# HACKER

Season 05 / Issue 08
τεύχος 056 - Nordostbahnhof Edition



# Πώς ξεκινούν τα PC & πώς φορτώνει το Linux [*]

- Πρακτική εξάσκηση με την SQLite
- Δημιουργία OpenBSD-based router
- Resolving για τα FQDNs στο LAN σας
- Ασφαλέστερο DNS με χρήση του DNSCrypt
- Ασφαλής προσάρτηση απομακρυσμένων καταλόγων μέσω SSH

**[*] Επίσης: Γνωρίστε το Systemd & αναβαθμίστε σημαντικά τις γνώσεις σας περί Linux!**

*all rights reversed*

# HACKER

# #dH056 | Περιεχόμενα




| | |
|---|---|
| **Εκδότης / Διευθυντής** | Χρήστος Βαρελάς |
| **Αρχισυντάκτης** | Παναγιώτης Βαρελάς |
| **Σύμβουλοι έκδοσης** | Νίκος Μουρατίδης, Βούλα Παυλίδου |
| **Σύνταξη** | Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Πάνος Γεωργιάδης, Γιώργος Γιάννου, Μάριος Καραγιαννόπουλος, Γιάννης Κρομμύδας, Πέτρος Κυλαδίτης, Μέσος Παπαδόπουλος (TM), Χρίστος Τόμπρας, @arkoudos |
| **Γραμματική επιμέλεια** | Βούλα Παυλίδου |
| **Καλλιτεχνική Επιμέλεια** | Πέτρος Φιλιππίδης |

Το Το ψηφιακό περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations. Η Parabing Creations έχει την έδρα της στην Καλαμαριά Θεσσαλονίκης. Ο δικτυακός τόπος του περιοδικού είναι στο http://deltahacker.gr. Το email επικοινωνίας είναι το talk2us@deltahacker.gr και ειδικά για τις συνδρομές είναι το subscriptions@deltahacker.gr. Οι γενικές πληροφορίες για τις συνδρομές παρατίθενται στο http://deltahacker.gr/pdf12order και από την ίδια σελίδα γίνονται οι παραγγελίες συνδρομών.

# Η δεύτερη καλύτερη εποχή

*του Χρήστου Βαρελά*

Φίλες και φίλοι,

- Τα πράγματα δεν δείχνουν καλά. Πολλά είναι τα στραβά και τα παράξενα που συμβαίνουν τον τελευταίο καιρό, ακόμη περισσότερα τ' ανησυχητικά σημάδια.
- Το περασμένο Σάββατο, π.χ., ταξιτζής με διαβεβαίωσε ότι *και* φέτος θα κάνουμε Χριστούγεννα με 15 βαθμούς και καθαρούς ουρανούς.
- Δεν είναι ότι του συγκεκριμένου ταξιτζή του άρεσε το κρύο και χαιρόταν που *δεν* θα έχουμε. Μου το ξεκαθάρισε ο άνθρωπος. "Όπως και να το κάνουμε όμως", πρόσθεσε με νόημα παρεμβάλλοντας δραματική παύση τριών δευτερολέπτων, "δεν είναι νορμάλ αυτές οι θερμοκρασίες για Χριστούγεννα".
- Μη βιαστείτε να σκεφτείτε κάτι σαν "ταξιτζής είναι, ό,τι θέλει λέει". Θα θυμόσαστε τι χειμώνα γιαλαντζί είχαμε πέρσι, έτσι δεν είναι;
- Δεν ξέρω, μακάρι να κάνω λάθος, αυτή τη φορά όμως τείνω να συμφωνήσω με τον ταξιτζή. Τα πρόωρα σημάδια του καιρού φανερώνουν ότι η κατάσταση δεν θα εξελιχθεί καλά – και για να είμαστε ειλικρινείς δείχνει να πηγαίνει από το κακό στο χειρότερο.
- Το ίδιο ισχύει και με μεγάλο μέρος του κόσμου.
- Δεν αναφέρομαι στον πλανήτη μας και στους κατοίκους του, αλλά στη χώρα μας και στους συμπολίτες μας.
- Δεν πάνε καλά κάποιοι άνθρωποι – ή τουλάχιστον σημαντικό ποσοστό κάποιων ανθρώπων.
- Το πιστεύω αυτό και το γράφω δημόσια κι ας κοστίσει στο περιοδικό συνδρομητές.
- Αναλογιστείτε, π.χ., ότι αρκετοί Εκεί Έξω™ πιστεύουν ακράδαντα ότι ο αναπληρωτής υπουργός υγείας "καλά τους τα λέει, έτσι θέλουν αυτοί". Μερικοί το πάνε και λίγο παραπέρα: "δέκα σαν αυτόν να 'χε η κυβέρνηση και θα τους είχαμε βάλει όλους στη θέση τους".
- Ποιους θα είχαμε βάλει στη θέση τους;

- Μα, όλους τους εκπροσώπους της εγχώριας διαπλοκής.
- Νομίζω –και στοιχηματίζω πως δεν κάνω λάθος στον υπολογισμό μου– ότι αν είχαμε δεκαπέντε σαν αυτόν θα 'χαμε βάλει και τους δανειστές στη θέση τους. Το τι σημαίνει αυτό, θαρρώ πως το γνωρίζετε: εμείς (στρωτή) λύρα, εκείνοι (άτσαλο) χορό.
- Παράξενα πράγματα, αλλά για ορισμένους όχι και τόσο.
- Σαν να μη μας έφτανε ο αναλογικός κόσμος, έχουμε ν' ανησυχούμε και για τον ψηφιακό ή εικονικό κόσμο του ίντερνετ. Να πω την αλήθεια μου, δεν μ' αρέσει να τον αποκαλώ ούτε έτσι ούτε αλλιώς. Το ίντερνετ δεν είναι ξεκομμένο από την καθημερινότητα και τις ζωές μας. Ίσα ίσα που αισθανόμαστε το αντίθετο, ότι δηλαδή ξαφνικά *εμείς* χάνουμε επαφή με τα πάντα όλα, όποτε τυχαίνει και είμαστε offline για μερικές μέρες (χτύπα ξύλο).
- Τέλος πάντων, δεν θα ήθελα να κάνω τώρα συζήτηση για το ίντερνετ και για τη σημασία του στις ζωές μας. Υπάρχουν άλλωστε άλλοι, σπουδαίοι διανοητές κι εξέχουσες προσωπικότητες, οι οποίοι έχουν να πουν πολλά επί του θέματος – και μάλιστα κάποιοι εξ αυτών έχουν αμολήσει κι ολόκληρα βιβλία με υπέροχα core dumps της συσσωρευμένης από τα χρόνια σοφίας τους.
- Αυτό που πράγματι θα ήθελα να κάνω είναι μια σύντομη αναφορά σε πρόσφατη επίθεση κατά του δικτύου ανωνυμίας του Tor.
- Θέλεις να σερφάρεις στο web ανώνυμα, π.χ., για να τσεκάρεις τι κάνουν οι ανταγωνιστές σου; Ζεις σε καταπιεστικό καθεστώς και δραστηριοποιείσαι πολιτικά, προσπαθώντας την ίδια στιγμή να καλύψεις τα ίχνη σου; Είσαι μήπως Τιτάνας στις αποκαλύψεις και ξεσκεπάζεις τη μία απατεωνιά μετά την άλλη, εκθέτοντας ακόμα και σημαίνοντα πρόσωπα ανεξαρτήτως της θέσης και της ισχύος τους; Θέλεις να εξερευνήσεις το λεγόμενο Dark Web, για λόγους που μόνο να φανταστούμε θα μπορούσαμε;
- Σε κάθε περίπτωση, δεν βγαίνεις στο "κανονικό" ίντερνετ απευθείας αλλά μπαίνεις πρώτα στο δίκτυο ανωνυμίας του Tor. Ουσιαστικά, εκεί μέσα διατρέχεις τυχαία παραγόμενες διαδρομές από host σε host, όπου ο καθένας εξ αυτών γνωρίζει μόνο το δημόσιο IP του προηγουμένου. Τελικά, κάποια στιγμή φτάνεις σε έναν ξεχωριστής σημασίας host, ο οποίος ονομάζεται exit relay, και μετά, απ' αυτόν, βγαίνεις στο κανονικό ίντερνετ. Ο server-προορισμός στον οποίο καταλήγεις, είναι σε θέση να καταγράψει μόνο το IP του exit relay. Η βήμα προς βήμα οπισθοδρόμηση, από τον exit relay έως τον υπολογιστή μας, είναι εξαιρετικά δύσκολη – αν όχι πρακτικά αδύνατη. Έτσι, θεωρητικά τουλάχιστον, η δικτυακή μας ταυτότητα προστατεύεται.

- Ας το δούμε όλα αυτό και σχηματικά. Μεταξύ του υπολογιστή μας και των απομακρυσμένων servers στους οποίους συνδεόμαστε, υπάρχει ένα νέφος από hosts. Κάποιος που παρατηρεί τα δικτυακά μας πακέτα να βγαίνουν από ένα σημείο του νέφους, δικαιούμαστε να λέμε ότι βρίσκεται σε προνομιακή θέση. Δεν πρόκειται για έναν απλό παίκτη, είναι νομίζουμε κατανοητό αυτό. Παρά όμως τη θέση ισχύος που κατέχει, δεν του είναι καθόλου εύκολο να μπει μέσα στο νέφος και να βγει από το σημείο που εμείς συνδεθήκαμε.
- Επαναλαμβάνουμε ότι όλα τα παραπάνω ισχύουν στη θεωρία. Στην πράξη, τα πράγματα δεν είναι πάντα τόσο ρόδινα. Ίσως, π.χ., η δημόσια διεύθυνση IP που έχουμε από τον ISP –πρόκειται για εκείνη που πάση θυσία θέλουμε να κρύβουμε– διαρρέει εξαιτίας κάποιου plugin ή addon. Ακόμα χειρότερα, δεν αποκλείεται να γνωστοποιείται σε τρίτους από malware που κατοικοεδρεύει στον υπολογιστή μας.
- Υποθέστε τώρα ότι είμαστε απαλλαγμένοι από κάθε είδους plugin/addon/malware και γενικά το δημόσιο IP μας δεν διαρρέει εξαιτίας –κακόβουλου ή όχι– λογισμικού.
- Εδώ αρχίζουν τ' ανησυχητικά, σύμφωνα και με την είδηση που ανέφερα πριν λίγο (Google it). Πράγματι, ένας προνομιακός παίκτης εξακολουθεί να 'ναι σε θέση ν' αποκαλύψει την πραγματική διεύθυνση IP που έχουμε. Πώς μπορεί να το κάνει αυτό; Γνωρίζει μήπως τίποτε μυστικά περάσματα μέσα στο νέφος των hosts του Tor; Μπα, δεν υπάρχουν τέτοια περάσματα. Ωραία. Τότε πώς αλλιώς βρίσκει το IP μας;
- Οι ερευνητές που δημοσίευσαν τη σχετική εργασία (https://nymity.ch/dns-traffic-correlation/tor-dns.pdf) εξηγούν ότι καταφέρνουν ό,τι καταφέρνουν παρατηρώντας τη δραστηριότητα των nameservers. Ισχυρίζονται ότι παρατηρώντας κάποιος το DNS traffic και σε συνδυασμό με γνωστές τεχνικές fingerprinting, εξαπολύει επιτυχημένες επιθέσεις αναγνώρισης κι αποκάλυψης.
- Ποιοι όμως είναι ικανοί για να πραγματοποιούν τέτοιες διαβολεμένες επιθέσεις; Ευτυχώς, όχι πολλοί. Ο επιτιθέμενος, βλέπετε, οφείλει να βρίσκεται σε θέση ώστε να παρατηρεί τόσο τα δικτυακά πακέτα που εισέρχονται στο νέφος του Tor, όσο κι εκείνα που εξέρχονται. Επιπλέον, η παρατήρηση από μόνη της δεν αρκεί και πρέπει να γίνονται και οι κατάλληλοι συσχετισμοί. Με δεδομένο ότι το νέφος του Tor απαρτίζεται από hosts που είναι διάσπαρτοι στον πλανήτη, καταλαβαίνετε ότι δεν μιλάμε για εύκολες επιχειρήσεις εδώ. Από την άλλη, όταν κάποιος μπαίνει στο στόχαστρο, τότε η κλίμακα της επιχείρησης ίσως και να μην έχει και τόσο σημασία.

- Ας συζητήσουμε λίγο πιο συγκεκριμένα για τους παίκτες που πράγματι θα μπορούσαν να ξεκινήσουν μια τέτοια επίθεση συσχέτισης παρατηρήσεων. Το δίκτυο του Tor είναι αρκετά περίπλοκο και αποκεντρωμένο, όμως δεν ισχύει το ίδιο για το ευρύτερο δίκτυο εντός του οποίου βρίσκεται το νέφος του Tor. Δείτε, π.χ., τι γίνεται με τους δημόσιους και δωρεάν nameservers στο ίντερνετ. Μπορείτε να μαντέψετε σε ποια εταιρεία ανήκουν οι δημοφιλέστεροι εξ αυτών;
- Σωστά. Στη Google. Απειραριθμήσιμο πλήθος χρηστών και διαχειριστών ανά τον πλανήτη, χρησιμοποιούν τους λαοφιλείς 8.8.8.8 και 8.8.4.4. Σύμφωνα μάλιστα και με την εργασία που συζητάμε, το ένα τρίτο των DNS requests που ξεκινούν από exit relays καταλήγουν σε nameservers της Google.
- Τι σημαίνει αυτό; Θεωρητικά τουλάχιστον, σημαίνει ότι κάποια ομάδα μέσα από τη Google θα μπορούσε να βρεθεί σε προνομιακή θέση για επιθέσεις συσχετισμού. Τέτοιες επιθέσεις μάλιστα διευκολύνονται σημαντικά όταν το υποψήφιο θύμα επισκέπτεται μη-δημοφιλείς δικτυακούς τόπους, αφού τα σχετικά DNS records ξεχωρίζουν ευκολότερα. Σκεφτείτε τώρα ότι ένας χρήστης του Tor που καταφεύγει στο εν λόγω δίκτυο ανωνυμίας όχι περιστασιακά αλλά συχνά και για σοβαρούς λόγους, μάλλον δεν θα καταλήγει σε μεγάλα, γνωστά sites.
- Έχουμε άραγε εμείς, οι απλοί και ταπεινοί χρήστες του Tor, λόγους για ν' ανησυχούμε; Όχι ακόμα, σύμφωνα με τους ερευνητές. Πρόκειται ν' αντιδράσει η ομάδα ανάπτυξης του Tor, εν όψει των πρόσφατων ανακαλύψεων κι αποκαλύψεων; Ευτυχώς έχει ήδη αρχίσει να διερευνά μεθόδους, ώστε το fingerprinting να καθίσταται ακόμη πιο δύσκολο.
- Πέρα από το τεχνικό κι ερευνητικό ενδιαφέρον που παρουσιάζουν οι εργασίες περί των αντοχών του Tor, εσείς πώς τα βλέπετε όλα αυτά; Πρόκειται μήπως ν' αλλάξετε ή να αναπροσαρμόσετε τις πρακτικές υπέρ της προστασίας της ανωνυμίας σας; Εντάξει, πιθανώς δεν χρησιμοποιείτε το Tor –τουλάχιστον όχι για πολύ σοβαρούς λόγους–, ίσως όμως τώρα σκέφτεστε ότι τίποτα δεν είναι 100% ασφαλές. Το ίδιο βεβαίως θα ισχύει και γι' αυτές τις συνδέσεις VPN που συχνά-πυκνά κάνετε (και ξέρετε εσείς γιατί).
- Δεν ξέρουμε πώς τα βλέπετε όλα αυτά, θα χαρούμε όμως να μας πείτε τη γνώμη σας.
- Όπως και να 'χει, χαρείτε το νέο τεύχος του deltaHacker – βεβαίως και τη δεύτερη καλύτερη εποχή του χρόνου.




# σνέιλ μέιλ

Αγαπητοί μου,

Συνειδητοποιώ πως έχει περάσει πολύς καιρός από την προηγούμενή μου επιστολή.

Μου το θύμισε η μητέρα της εξαδέλφης σήμερα το πρωί.

Μόλις είχα επιστρέψει, και πήγαινα προς τα μέσα σέρνοντας τη βαλίτσα.

"Τελικά αυτή τη γραφομηχανή τη χρειάζεσαι ή την έχεις αφήσει μεσ' στη μέση για να σκονίζεται;"

Φωνή της συνείδησης, ένα πράμα. Το έπιασα το υπονοούμενο, που λέτε, και να 'μαι.

Η αλήθεια είναι πως από την τελευταία φορά που σας έγραφα έχουν γίνει ένα σωρό πράγματα.

Το βασικότερο, βέβαια, αυτό που μας έβγαλε εντελώς από το πρόγραμμά μας, ήταν το Brexit.

Ο ανιψιός δηλαδή.

Το καημένο το παιδί τα 'παιξε. Τον πήρε από κάτω.

Ό,τι και να του λέγαμε, τίποτα, χαμένο πήγαινε. Αφού είδαμε και αποείδαμε μέσω σκάιπ και δεν γινόταν τίποτα, πήρα τη Μαιρούλα και πήγαμε να τον επισκεφτούμε.

Η Κυρία Καίτη έκατσε Αθήνα να κάνει παρέα στη μητέρα της εξαδέλφης, που είναι Ινδουίστρια.

Κάτσαμε Λονδίνο δέκα μέρες. Όσο χρειαζόταν για να δροσίσουμε λίγο το κοκαλάκι μας και να πάρουμε μιαν ιδέα της κατάστασης.

Αυτό για τη δροσιά το λέω και το εννοώ. Πρώτη βδομάδα του Ιουλίου, και στο Λονδίνο μόνο που δε χιόνιζε. Ό,τι χρειαζόμασταν για ένα reality check.

Μετά έφτιαξε λίγο, κι αυτές τις μέρες έχει καύσωνα.

Αλλά δεν είμαστε πλέον Λονδίνο. Κάτσαμε, όπως λέω, καμιά δεκαριά μέρες και μετά επιστρέψαμε. Πήγαμε κατευθείαν στο Πόρτο Γερμενό, όπου μείναμε άλλο τόσο. Γι αυτό και βρέθηκα χωρίς τη γραφομηχανή. Δηλαδή τι γραφομηχανή, εδώ η Μαιρούλα βρέθηκε χωρίς άιΦον γιατί είχε ξεχάσει τον φορτιστή της.

Επιστρέψαμε με το πρωινό λεωφορείο σήμερα. Μόλις προ ολίγου.

Αφού φορτίσανε τα κινητά τους, ο ανιψιός με τη Μαιρούλα βγήκανε για μια βόλτα στο κέντρο να πιάσουν Πόκεμον, κι εγώ έκατσα για να σας γράψω.

Αναρωτιόμουν από πού να το πιάσω.

Θα θυμάστε ίσως τις ζοφερές μου σκέψεις για το πώς θα πάνε τα πράγματα αν επικρατούσε η αφροσύνη και αποφάσιζαν οι Εγγλέζοι να ψηφίσουν για Brexit.

Ε, λοιπόν, και λίγα σας έγραφα.

Αυτό που δίσταζα να παραδεχτώ είχε να κάνει με τη δυναμική του πράγματος.

Νόμιζα --ή, για να το πω καλύτερα, είχα την ελπίδα-- πως οι αιτίες που θα έκαναν κάποιον να ψηφίσει για Brexit θα ήταν συγκυριακές: οι ψεύτικες υποσχέσεις του τάδε λαϊκιστή πολιτικού, αυτή ή εκείνη η αιτία δυσαρέσκειας κ.λπ. Ήλπιζα, δηλαδή, πως μπροστά στην κάλπη κάποιοι θα το ξανασκέφτονταν και θα επικρατούσε η "λογική" --ή έστω αυτό που εγώ ονομάζω "λογική".

Αυτή ήταν η ελπίδα, όχι μόνο η δική μου, αλλά και πολλών άλλων. Οι περισσότεροι έτσι πίστευαν πως θα γίνει. Απ' ό,τι μου έλεγε και ο ανιψιός, δικοί του γνωστοί και φίλοι παρέμεναν χαλαροί και ψύχραιμοι μέχρι και την τελευταία στιγμή. Μερικοί απ' αυτούς, μάλιστα, δεν θα πήγαιναν καν να ψηφίσουν. Τόσο σίγουροι ένοιωθαν.

Υπό την έννοια αυτή, το αποτέλεσμα του βρετανικού δημοψηφίσματος αποτελεί το καλύτερο wake up call.

Κανείς δεν το περίμενε. Όλοι "έπεσαν από τα σύννεφα". Ακόμα και όσοι πάλευαν για το Brexit.

Εξ ου και το μούδιασμα όλων, αμέσως μετά την ανακοίνωση των αποτελεσμάτων.

Το βασικότερο παράδοξο αυτού του δημοψηφίσματος δεν ήταν η απογοήτευση των χαμένων. Αυτή ήταν προφανώς αναμενόμενη. Όποιος και να ήταν ο χαμένος, απογοητευμένος θα ήταν.

Αυτό που έμοιαζε εντελώς παράταιρο, σ' εμένα τουλάχιστον, ήταν η ανησυχία των "νικητών".

Την έβλεπα καθαρά αυτές τις λίγες μέρες που περάσαμε στο Λονδίνο.

Όχι όλων των νικητών, βεβαίως. Υπήρχαν και οι χαρούμενοι νικητές. Οι πειραγμένοι, τα χρυσά αβγά του ξενοφοβικού φιδιού, που ακούσανε για Brexit κι αρχίσανε τους τραμπουκισμούς.

Οι άλλοι όμως, οι υπόλοιποι, παρέμεναν σιωπηλοί, τρομοκρατημένοι --το έβλεπες-- απέναντι στη διστακτική απειλή που εκπέμπει η τυφλή σιγουριά ενός πλήθους, τη στιγμή που συνειδητοποιεί ξαφνικά ότι είναι η πλειοψηφία.

Αυτό ήταν που τρομοκράτησε κι εμένα.

Το Brexit θα προχωρήσει επειδή έτσι το θέλησε το πλήθος.

Η χώρα θα κάνει ένα ακόμα βήμα προς το γκρεμό, επειδή έτσι το θέλησε το πλήθος.

Η Ευρωπαϊκή Ένωση θα κάνει ένα ακόμα βήμα προς την αυτοακύρωσή της, επειδή έτσι το θέλησε το πλήθος.

Η Ευρώπη θα ξαναγίνει επικίνδυνο μέρος, επειδή έτσι το θέλησε το πλήθος.

Ξέρω, σας ξενίζει αυτό που λέω. Σας ακούγεται, είμαι σίγουρος, αντιδημοκρατικό. Αυταρχικό. Επικίνδυνο.

Πράγματι, έτσι ακούγεται. Αλλά να, μην πάμε μακριά. Ρίξτε μια ματιά στη γειτονική μας Τουρκία.



Πραξικόπημα, απειλείται η Δημοκρατία, και τώρα το πλήθος ζητάει κρεμάλες.

Το φιλοδημοκρατικό πλήθος ζητάει κρεμάλες για τους εχθρούς της δημοκρατίας.

Τη βλέπετε την αντίφαση; Όχι;

Μα, αν Δημοκρατία είναι το σύστημα εκείνο που σου επιτρέπει να έχεις γνώμη, δεν εξυπακούεται πως πρέπει επίσης να σου επιτρέπει να _αλλάζεις_ γνώμη;

Αλλά πώς μπορείς να αλλάξεις γνώμη, αν το διακύβευμα έχει περάσει από την κρεμάλα;

Δεν σας λέω κάτι καινούργιο. Τα ξανάγραφα αυτά το Δεκέμβριο του 2013 (τεύχος 027).

"Δηλαδή ποιος σου είπε", ρωτούσα τότε την εξαδέλφη, "πως τα θεμέλια της δημοκρατίας είναι δημοκρατικά; Καθόλου δημοκρατικά δεν είναι."

"Κοίτα", συνέχιζα, "πάρε για παράδειγμα τις εκλογές. Εκλογές γίνονται, είτε το θέλεις είτε δεν το θέλεις. Κανείς δεν σε ρωτάει αν σ' αρέσει να γίνονται εκλογές και κανείς δεν πρόκειται να σε ρωτήσει. Εκλογές γίνονται γιατί το λέει το Σύνταγμα. Τέρμα. Το Σύνταγμα δεν το έχουμε βάλει εκεί για να το συζητάμε κάθε φορά. Αποφασίστηκε, ισχύει, πάει τέλειωσε. Να το αλλάξεις δε γίνεται με το έτσι θέλω, θέλει διαδικασίες, θέλει δουλειά. Και κάποια πράγματα δεν αλλάζουν με τίποτα."

"Φαντάσου", της έλεγα, "να γινόταν αύριο δημοψήφισμα με το ερώτημα αν θέλουμε να γίνονται δημοψηφίσματα στο μέλλον. Φαντάσου τώρα ότι κάπως μας ήρθε κι αποφασίσαμε όλοι σύσσωμοι ότι δεν θέλουμε. Τα βαρεθήκαμε. Τελεία. Δημοψηφίσματα τέρμα. Φαντάσου τώρα ότι περνάνε είκοσι χρόνια χωρίς δημοψηφίσματα και αρχίζουμε να το ξανασκεφτόμαστε. Ότι έχουν αρχίσει να μας ψιλολείπουν. Πώς θα το αποφασίσουμε να τα επαναφέρουμε, αν πλέον δεν μπορούμε να θέσουμε το ερώτημα σε δημοψήφισμα; Βλέπεις τι εννοώ; Βλέπεις την αντίφαση;"

Αυτά τα έλεγα το Δεκέμβριο του 2013. Με αφορμή τη σύλληψη, τότε, του συνόλου σχεδόν της ηγεσίας της Χρυσής Αυγής. Τα παλιά καλά χρόνια. Τα γεμάτα νοσταλγία.

Από τότε σαν να 'χουν περάσει αιώνες.

Περασμένα ξεχασμένα.

Όλη η Χρυσή Αυγή είναι ξανά έξω, αλλά η δίκη προχωρά.

Αναμένεται να ολοκληρωθεί πανηγυρικά just in time για τις εκδηλώσεις με αφορμή τα 100 χρόνια από την έναρξη του Β' Παγκοσμίου Πολέμου.

Αλλά μην ξεφεύγω από το θέμα μου. Η εμπιστοσύνη μου στην Ελληνική δικαιοσύνη είναι απόλυτη.

(not)

Για τη Δημοκρατία, όμως, τα ίδια θα έγραφα και σήμερα.

Αγαπητοί μου,

Τα θεμέλια της δημοκρατίας δεν μπορεί παρά να είναι αντιδημοκρατικά.

Η Δημοκρατία δεν είναι ούτε φυσική, ούτε αυτονόητη κατάσταση των ανθρωπίνων κοινωνιών. Είναι ένα αιματηρά κατακτημένο πλαίσιο περιορισμού του τι μπορεί να κάνει ο εκάστοτε ισχυρός στον εκάστοτε ανίσχυρο.

Η Δημοκρατία δεν είναι εκεί για να σου λέει τι μπορείς να κάνεις. Είναι για να σου λέει τι *δεν* μπορείς να κάνεις.

Όσο το ξεχνάμε αυτό, όσο παρασυρόμαστε από την αμεσοδημοκρατική αποφασιστικότητα του κάθε προφήτη, τόσο πιο ζοφερά φαντάζουν τα σύννεφα που έχουν ήδη αρχίσει να μαζεύονται.

Και τόσο πιο έρημοι οι δρόμοι όταν δύει ο ήλιος

Σας ασπάζομαι,
Θείος Ακάκιος












The IEEE Global Engineering Education Conference (EDUCON) 2017 is the eighth in a series of conferences that rotate among central locations in IEEE Region 8 (Europe, Middle East and North Africa). EDUCON is the flagship conference of the IEEE Education Society.

The theme of EDUCON 2017 conference is:

**Challenging the transition from the classic to the emerging in the Engineering Education**

www.educon-conference.org/educon2017

## Date and Venue

IEEE EDUCON 2017 will be organized by the University of Piraeus, the Hellenic Open University, the Hellenic Air Force Academy and the Technological Educational Institute of Athens, Greece. The event will be held in Athens, Greece April 26-28, 2017. Several pre-conference workshops will be held on April 25, 2017.

## Athens, Greece

Athens is the capital and largest city of Greece. Athens is one of the world's oldest cities, widely considered the cradle of Western civilization and the birthplace of democracy. Modern Athens is a large cosmopolitan metropolis and central to the economic, financial, industrial, maritime, political and cultural life in Greece. (http://www.greece-athens.com)

## Deadlines

| Date | Item |
|---|---|
| **Oct. 20, 2016** | Abstract submission |
| **Oct. 24, 2016** | Invitation to submit complete paper |
| **Nov. 28, 2016** | Complete paper submission<br>Proposals for Special Sessions, Panels and Workshops |
| **Dec. 21, 2016** | Notification of acceptance |
| **Feb. 06, 2017** | Author registration and payment,<br>Camera ready submission |
| **April 25, 2017** | Pre-conference workshops |
| **April 26-28, 2017** | EDUCON 2017 |

## Main Conference Topics

Papers in all areas of engineering education are invited with a particular emphasis on the theme of the conference and on:

- Entrepreneurship and Innovation in Engineering Education
- Serious Games and Game-Based Learning for Engineering Education
- Innovative Materials, Teaching and Learning Experiences in Engineering Education
- Engineering Education and Cultural Heritage
- Smart Education approaches

## Presentation Formats

Proposals for the EDUCON 2017 conference can be submitted in the following formats:

- Full papers: 8 pages in IEEE A4 format
- Short papers: 4 pages
- Works in Progress
- Special Sessions
- Panels and Workshops
- Posters

## Organizers

**University of Piraeus**
www.unipi.gr

**Hellenic Open University**
www.eap.gr

**Hellenic Air Force Academy**
https://www.haf.gr/career/academies/haf-academy/

**Technological Educational Institute of Athens**
www.teiath.gr

**Piraeus University of Applied Sciences**
www.teipir.gr

ΕΛΛΗΝΙΚΟ ΑΝΟΙΚΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΤΕΙ ΑΘΗΝΑΣ

UNIVERSITY OF PIRAEUS RESEARCH CENTER

**Skill:** Intermediate
**Tags:** OpenBSD, DHCP, DNS, dnsmasq, DNSCrypt, pf

# Οικιακός OpenBSD router, με DNSCrypt & resolving τοπικών FQDNs

**Εμπιστευόσαστε την ασφάλεια του τοπικού σας δικτύου στο modem/router που έχετε πάρει από τον ISP; Αν η απάντηση που δίνετε στο ερώτημα είναι, το πολύ, *μουδιασμένα* καταφατική, τότε βρισκόσαστε στο σωστό μέρος.**

***του Χ.Β.***

Αφορμή για το παρόν άρθρο αποτέλεσε email που είχαμε λάβει αρκετό καιρό πριν από αναγνώστη και φίλο, ο οποίος τα τελευταία χρόνια ζει κι εργάζεται στο εξωτερικό. Δεν πρόκειται για μεμονωμένο περιστατικό, μας έγραφε, αλλά μάλλον για τάση: τα modem/routers που παρέχουν οι ISPs είναι περισσότερο modems, παρά οτιδήποτε άλλο. Ακόμη και στις περιπτώσεις που ο κάτοχος έχει δυνατότητα για τροποποίηση των κανόνων του firewall, ρυθμίσεις port forwarding, DMZ κ.λπ., η απλοϊκή λογική του web panel που παρουσιάζει η συσκευή μόνο εμπιστοσύνη δεν εμπνέει.

Δεν χρειάζεται να συζητήσουμε πολύ ώστε να καταλάβουμε γιατί οι υποψιασμένοι χρήστες τέτοιων modems αναζητούν εναλλακτικές λύσεις. Αρκεί μόνο ν' αναφέρουμε ότι οι εν λόγω συσκευές αναβαθμίζονται αυτόματα, χωρίς να υπάρχει επιλογή για απενεργοποίηση της συγκεκριμένης δυνατότητας. Αμέσως αμέσως, λοιπόν, τίθενται τέσσερα άβολα ερωτήματα:

**α)** Πόσα προβλήματα διορθώνει ένα update και πόσα νέα εισάγει;

**β)** Πόσο εμπιστευόμαστε τους υπευθύνους για τα updates;

**γ)** Οι όποιες ρυθμίσεις/αλλαγές έχουμε κάνει, συνεχίζουν άραγε να ισχύουν και μετά από κάθε αυτόματη αναβάθμιση;

**δ)** Μήπως μετά από αναβάθμιση η συσκευή αποκτά δυνατότητες που βολεύουν τον ISP και τους συνεργάτες του, παραβιάζουν όμως την ιδιωτικότητά μας;

Κοιτάξτε, απαντήσεις δεν έχουμε – εμείς απλά τα ερωτήματα θέτουμε :P Σημειώνουμε επίσης ότι παρόμοια συμπεριφορά και features έχουν και τα modems που δίνουν κι ελληνικοί ISPs.

Η λύση που σκέφτηκε ο φίλος μας είναι ν' αναλάβει ο ίδιος την ασφάλεια του οικιακού του δικτύου. Πιο συγκεκριμένα, αποφάσισε να στήσει ένα δικό του firewall/router *ακριβώς* πίσω από το modem, και μάλιστα να απενεργοποιήσει εντελώς το firewall που έχει το προαναφερθέν. Όλες οι συσκευές του (desktop, game console, laptop, smartphones, tablet, Chromecast, εκτυπωτής/σαρωτής και δεν συμμαζεύεται) θα βρίσκονται πίσω από το firewall/router και θα προστατεύονται απ' αυτό. Έτσι, ο ISP θα μπορεί να κάνει ό,τι θέλει με το modem του, τη στιγμή που ο πελάτης δεν θα μπορούσε να νοιαστεί λιγότερο. Κι αν ο ISP αποφασίσει να μάθει τι παίζει στο τοπικό δίκτυο του πελάτη, το πολύ πολύ να δει έναν αποφασιμένο router να του επιστρέφει ένα βλοσυρό βλέμμα. Το μόνο κακό που ίσως μπορέσει να κάνει ο ISP –άθελά του, εννοείται– είναι να εφαρμόσει ελαττωματικό firmware upgrade και να αχρηστεύσει το modem του πελάτη. Αυτό δεν το αναφέρουμε ως θεωρητικό ενδεχόμενο: Έχει συμβεί *τουλάχιστον* στον αναγνώστη μας και η λύση σε μια τέτοια περίπτωση είναι τηλέφωνο στο tech support, προσευχόμενος να πέσεις σε υπάλληλο που καταλαβαίνει δυο-τρεις αγγλικές λέξεις μπας και βγάλεις άκρη και σου στείλουν νέο router γρήγορα. (Το πιθανότερο είναι ν' ακούσεις κάτι σαν "ein Kollege wird Sie am Montag anrufen" ή ακόμα και κάτι που θα μοιάζει με "a samstarfsmaður mun hringja í þig á Mánudagur". Αλλά εντάξει, κάποια στιγμή θα σου στείλουν νέα συσκευή.)

## Λειτουργικό σύστημα και hardware

Για το OS του firewall/router αρχικά προτείναμε στον φίλο το pfSense, το οποίο χρησιμοποιούμε για πολύ καιρό και το εμπιστευόμαστε (https://deltahacker.gr/deltacast-s01e07). Ο ευκολότερος τρόπος για να έχεις pfSense, είναι να το εγκαταστήσεις σε ένα PC που σου περισσεύει. Μετά όμως ίσως υπάρξουν θέματα με τον θόρυβο, ο λογαριασμός του ηλεκτρικού θα 'ναι κάπως αυξημένος, κατά περίπτωση υπάρχει το ζήτημα της διαχείρισης χώρου – για να μην αναφερθούμε και στο χαμηλό WAF. Σε σχετική εξάλλου ιστοσελίδα του επίσημου δικτυακού τόπου του pfSense, παρατίθενται συγκεκριμένες προτάσεις εξειδικευμένου hardware (https://www.pfsense.org/hardware). Άριστη επιλογή για ένα οικιακό δίκτυο φαίνεται ν' αποτελεί το SG-1000 microFirewall, το οποίο αναμένεται από το Νοέμβριο του 2016 (https://netgate.com/products/sg-1000.html).

Τελικά ο αναγνώστης μας αποφάσισε να γυρίσει την πλάτη στο pfSense και να στραφεί στο OpenBSD (https://www.openbsd.org). Το σκεπτικό του ήταν ότι αφενός δεν τον ενδιαφέρουν τα πολλά features που έχει το pfSense, αφετέρου έψαχνε και μια δικαιολογία για να δοκιμάσει ένα λειτουργικό στο οποίο η ασφάλεια πάντα ήταν – και συνεχίζει να είναι– το άλφα και το ωμέγα. Όσον αφορά στο hardware για έναν OpenBSD-based home router, μια καλή επιλογή είναι το RCC-DFF 2220 της Netgate (http://store.netgate.com/ADI/RCC-DFF-2220.aspx). Είναι μικρό, fanless, οικονομικό στην κατανάλωση και με σχετικά δυνατά χαρακτηριστικά (dual-core Intel Atom CPU, 2GB RAM, 2 gigabit Ethernet ports). Μπορείτε επιπρόσθετα να του προσθέσετε δίσκο SSD (δεν είναι απαραίτητος), ενώ για wireless συνδέσεις επιλέγετε είτε κάποιο από

τα Access Points που προτείνει η Netgate είτε οποιοδήποτε άλλο της επιλογής σας. Να σημειώσουμε εξάλλου ότι μια εναλλακτική για hardware "χαμηλού προφίλ" αποτελούν τα APU2 boards της PC Engines (http://www.pcengines.ch/apu2.htm).

### Η παρουσίασή μας

Στη συνέχεια του παρόντος δείχνουμε, αναλυτικά και βήμα προς βήμα, πώς ρυθμίζουμε μια νέα εγκατάσταση του OpenBSD, ώστε το σύστημα να λειτουργεί ως firewall/router για ένα τοπικό δίκτυο. Στους πελάτες του θα παρέχει υπηρεσίες DHCP και DNS, ενώ θα μπορεί να κάνει και resolving τοπικών FQDNs (Fully Qualified Domain Names). Έτσι, αντί να γράφουμε τη διεύθυνση IP (π.χ., 192.168.42.195) ενός μηχανήματός μας στο οποίο θέλουμε να συνδεθούμε μέσω SSH, θα δίνουμε το πλήρες όνομά του εντός του domain (π.χ., einarben.colder.xyz, όπου υποθέσαμε ότι το hostname του απομακρυσμένου μηχανήματος είναι einarben και το domain είναι το colder.xyz). Ο nameserver του OpenBSD box θα είναι caching (https://deltahacker.gr/?p=14739). Τις διευθύνσεις IP που δεν έχει ήδη στην cache θα τις αναζητά μέσω του πρωτοκόλλου DNSCrypt, ως έξτρα μέτρο ασφαλείας. Το τοπικό δίκτυο, τέλος, θα προστατεύεται από το ικανότατο pf firewall.

Θεωρούμε εξάλλου ότι το hardware στο οποίο φιλοξενείται το OpenBSD έχει δύο κάρτες Ethernet: η μία για τη σύνδεση στο modem του ISP (WAN port), η αλλη για τη σύνδεση των πελατών (LAN port). Βεβαίως, στο LAN port θα έχουμε συνδεδεμένο κάποιο switch ή/και κάποιο wireless Access Point. Το θέμα είναι ότι, τελικά, οι συσκευές μας θα φτάνουν στον DIY router μέσω του LAN port – κι αυτό ακριβώς θέλουμε.

**Σημείωση.** Χάρη στις ιδιαίτερα χαμηλές απαιτήσεις που έχει το OpenBSD όσον αφορά στους πόρους συστήματος, άνετα μπορούμε να το έχουμε σε μηχανάκι με μόλις 64MB RAM. Μάλιστα ήδη διατηρούμε ένα VM παρόμοιων χαρακτηριστικών με guest OS το OpenBSD, σε ρόλο firewall/router για ένα από τα εικονικά μας εργαστήρια. Για τις ανάγκες της παρουσίασης που ακολουθεί εργαστήκαμε σε VMware VM με 64MB RAM, επεξεργαστή με δύο πυρήνες, δυναμικό δίσκο μεγέθους 16GB (το πραγματικό μέγεθος του αντίστοιχου αρχείου είναι μικρότερο από 700MB) και δύο network adapters: το WAN με NAT networking και το LAN με custom networking, ονόματι vmnet4. Οι πελάτες του OpenBSD firewall/router χρησιμοποιούσαν κι εκείνοι το vmnet4.

### Λήψη κι εγκατάσταση

Δουλεύουμε με την τελευταία έκδοση του OpenBSD, η οποία από τις αρχές του Σεπτεμβρίου είναι η 6.0. Το firewall/router δεν πρόκειται να 'χει πάνω από 4GB RAM, οπότε μεταξύ των εκδοχών 32bit κι 64bit, για επεξεργαστές Intel/AMD, επιλέγουμε την πρώτη. Αν φυσικά ο router σας πρόκειται να έχει 4GB RAM ή περισσότερα, στραφείτε στην εκδοχή 64bit. Σε κάθε περίπτωση, η σελίδα του download είναι στο

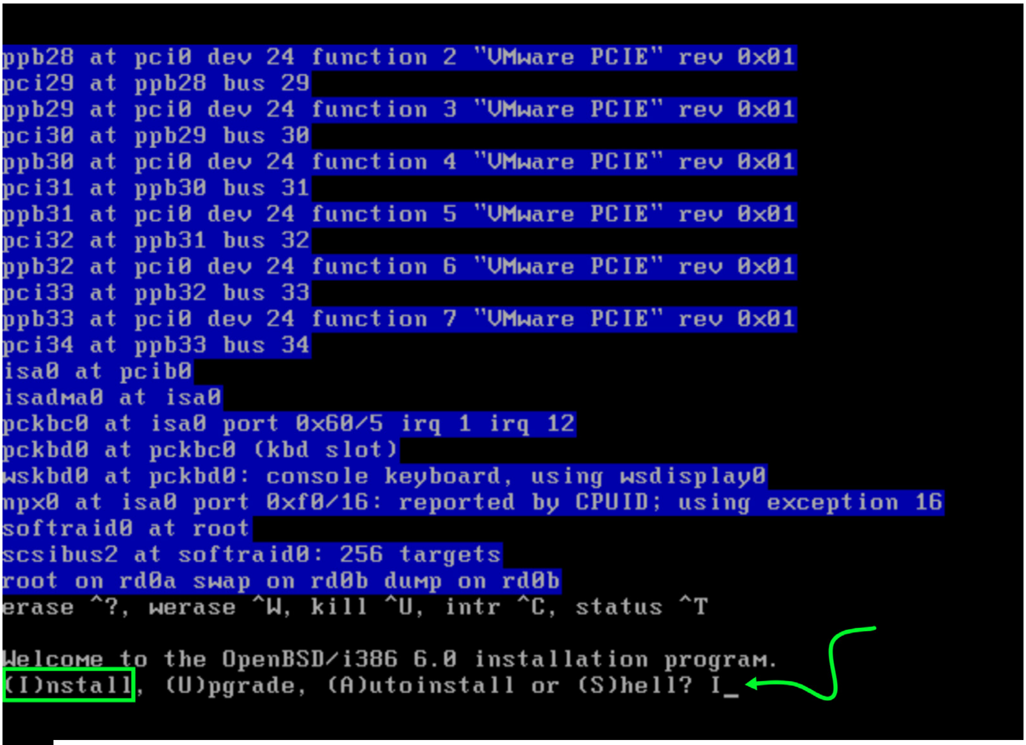
https://www.openbsd.org/ftp.html

κι εμείς κατεβάσαμε το ISO μας από το HTTP mirror στο Erlangen, στη διεύθυνση

http://openbsd.cs.fau.de/pub/OpenBSD

Όποιο κι αν είναι το mirror που θα επιλέξετε, μπείτε στο directory ονόματι 6.0, μετά σ' εκείνο της αρχιτεκτονικής που σας ενδιαφέρει και κατεβάστε το επιθυμητό ISO image. Αν, π.χ., θέλετε το image για πλήρη offline εγκατάσταση, πάρτε το αρχείο install60.iso (226,3MB). Αν πάλι προτιμάτε μικρότερο image και κατά τη διάρκεια της εγκατάστασης να κατέβουν μόνο τα απαραίτητα πακέτα, πάρτε το αρχείο cd60.iso (7,3MB).

Για την ίδια τη διαδικασία της εγκατάστασης του OpenBSD δεν χρειάζεται να πούμε πολλά, αφού ολοκληρώνεται εύκολα και με αποδοχή σχεδόν όλων των προεπιλογών. Αν όμως αυτή είναι η πρώτη σας φορά που εγκαθιστάτε OpenBSD, ίσως θελήσετε να παρακολουθήσετε το deltaCast s01e10 (https://deltahacker.gr/?p=9367). Δείτε επίσης μερικά χαρακτηριστικά screenshots που για λόγους πληρότητας παραθέτουμε, διαβάζοντας και τις αντίστοιχες περιγραφές.

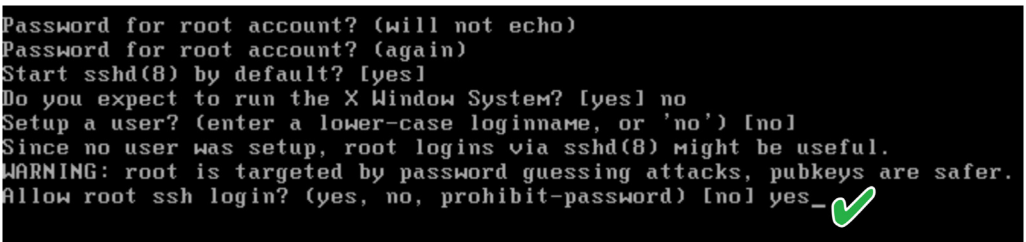



*O installer του OpenBSD τρέχει σε περιβάλλον κονσόλας κειμένου και προχωρά απευθύνοντάς μας μια σειρά από ερωτήσεις. Εδώ βλέπουμε την πρώτη ερώτηση αμέσως μετά την εκκίνησή του. Επειδή πρόκειται για καθαρή εγκατάσταση, μεταξύ των Install, Upgrade, Autoinstall και Shell, επιλέγουμε την πρώτη λειτουργία: απλά πατάμε το πλήκτρο [I] και μετά το [Enter].*



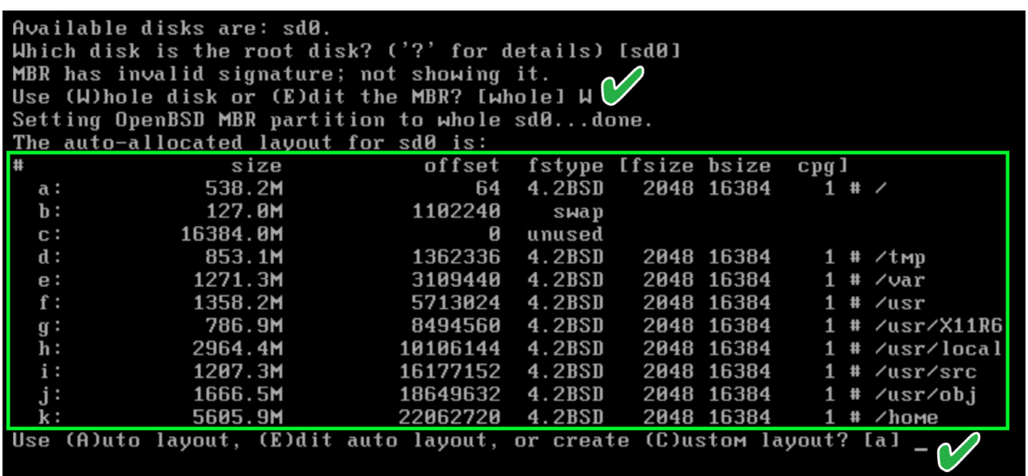

*Σε μία και μόνο οθόνη κάνουμε τις πιο σημαντικές επιλογές για το νέο σύστημα που τώρα εγκαθιστούμε. Αρχικά πληκτρολογούμε ένα hostname για το μηχάνημα (hauptbahnhof) – και για την ακρίβεια θέλουμε να ισχύει για το WAN interface (1) της συσκευής. Το προαναφερθέν αντιστοιχεί στο em0 interface του OpenBSD. Το firewall/router που στήνουμε δεν θέλουμε να παίρνει δυναμικά διεύθυνση IP από το modem/router, οπότε του δίνουμε ένα IP που να έχει νόημα για το δίκτυο ακριβώς πίσω από τη συσκευή του ISP. Για το netmask αφήνουμε την προεπιλογή (2). Δεν επιθυμούμε δικτύωση IPv6 για το em0 κι αμέσως προχωράμε στη ρύθμιση του em1, το οποίο είναι το LAN interface του router μας (3). Το βαφτίζουμε με ένα hostname της επιλογής μας (husavik), του δίνουμε μια έγκυρη διεύθυνση εσωτερικού δικτύου και, τέλος, ως netmask αφήνουμε το προτεινόμενο. Παρατηρήστε ότι στο em1 του παραδείγματος δώσαμε τη διεύθυνση 172.16.164.250, επομένως αυτή θα είναι η διεύθυνση του router (ή αλλιώς του gateway) για τις συσκευές πίσω από το OpenBSD firewall/router. Με βάση και το netmask, οι διευθύνσεις που θα έχουν οι συσκευές θα είναι της μορφής 172.16.164.*. Για άλλη μια φορά, δεν θέλουμε δικτύωση IPv6 για το em1. Στη συνέχεια πληκτρολογούμε τη διεύθυνση IP που έχει το modem/router του ISP (4). Πρόκειται για τη διεύθυνση gateway που έως τώρα έβλεπαν οι συσκευές μας και για το συγκεκριμένο παράδειγμα είναι η 192.168.201.2. Μετά δίνουμε ένα επιθυμητό domain name για το δίκτυο που θα εξυπηρετεί το OpenBSD firewall/router (5). Είμαστε περήφανοι κάτοχοι του colder.xyz, οπότε αυτό ακριβώς πληκτρολογήσαμε. Δίνουμε επίσης και μια διεύθυνση IP για τον DNS server, στον οποίο θα καταφεύγει ο nameserver του OpenBSD κάθε φορά που δεν θα βρίσκει τη ζητούμενη πληροφορία στην cache του. Προς το παρόν μπορεί να χρησιμοποιεί το modem/router του ISP για τις ανάγκες του, γι' αυτό κι εμείς πληκτρολογήσαμε τη διεύθυνση 192.168.201.2. Και μετά τα θέματα της δικτύωσης ορίζουμε ένα συνθηματικό (password) για το λογαριασμό του διαχειριστή*

*συστήματος (6), δηλαδή για τον root (το δίνουμε δύο φορές, για λόγους ασφαλείας). Επειδή θέλουμε να συνδεόμαστε στο firewall/router απομακρυσμένα και μέσω SSH, φροντίζουμε από τώρα ώστε η υπηρεσία του OpenSSH να 'ναι ενεργοποιημένη (7). Το router μας δεν χρειάζεται περιβάλλον γραφικών, οπότε στην ερώτηση περί X Window System απαντάμε αρνητικά. Επιλέγουμε τέλος να μη δημιουργήσουμε λογαριασμό απλού χρήστη, αφού μάλλον είναι περιττός για το συγκεκριμένο box.*



```
Password for root account? (will not echo)
Password for root account? (again)
Start sshd(8) by default? [yes]
Do you expect to run the X Window System? [yes] no
Setup a user? (enter a lower-case loginname, or 'no') [no]
Since no user was setup, root logins via sshd(8) might be useful.
WARNING: root is targeted by password guessing attacks, pubkeys are safer.
Allow root ssh login? (yes, no, prohibit-password) [no] yes_
```

*Αν και το SSH login απευθείας στο λογαριασμό του root και με χρήση password δεν είναι ό,τι πιο ασφαλές, προς το παρόν το επιτρέπουμε. Λίγο αργότερα θα φροντίσουμε ώστε να 'ναι δυνατό μόνο το key-based authentication. Ανά πάσα στιγμή, εξάλλου, αν θέλουμε φτιάχνουμε κι ένα λογαριασμό απλού χρήστη κι απαγορεύουμε το SSH στο λογαριασμό του root – μάλιστα ασχέτως της μεθόδου ταυτοποίησης.*



```
Available disks are: sd0.
Which disk is the root disk? ('?' for details) [sd0]
MBR has invalid signature; not showing it.
Use (W)hole disk or (E)dit the MBR? [whole] W
Setting OpenBSD MBR partition to whole sd0...done.
The auto-allocated layout for sd0 is:
#                size           offset  fstype [fsize bsize   cpg]
  a:           538.2M               64  4.2BSD   2048 16384     1 # /
  b:           127.0M          1102240    swap
  c:         16384.0M                0  unused
  d:           853.1M          1362336  4.2BSD   2048 16384     1 # /tmp
  e:          1271.3M          3109440  4.2BSD   2048 16384     1 # /var
  f:          1358.2M          5713024  4.2BSD   2048 16384     1 # /usr
  g:           786.9M          8494560  4.2BSD   2048 16384     1 # /usr/X11R6
  h:          2964.4M         10106144  4.2BSD   2048 16384     1 # /usr/local
  i:          1207.3M         16177152  4.2BSD   2048 16384     1 # /usr/src
  j:          1666.5M         18649632  4.2BSD   2048 16384     1 # /usr/obj
  k:          5605.9M         22062720  4.2BSD   2048 16384     1 # /home
Use (A)uto layout, (E)dit auto layout, or create (C)ustom layout? [a] _
```

*Έφτασε η στιγμή για τη διαμόρφωση του ενός και μοναδικού δίσκου του συστήματος, καθώς και για τη διευθέτηση των κατατμήσεων πάνω του. Δίνουμε όλο το δίσκο στο OpenBSD (επιλογή whole) και δεχόμαστε την πρόταση του installer για αυτόματη διευθέτηση κατατμήσεων (επιλογή auto layout).*

```
Let's install the sets!
Location of sets? (cd0 disk http or 'done') [cd0]
Pathname to the sets? (or 'done') [6.0/i386]

Select sets by entering a set name, a file name pattern or 'all'. De-select
sets by prepending a '-' to the set name, file name pattern or 'all'. Selected
sets are labelled '[X]'.
    [X] bsd            [X] base60.tgz     [X] game60.tgz     [X] xfont60.tgz
    [X] bsd.rd         [X] comp60.tgz     [X] xbase60.tgz    [X] xserv60.tgz
    [X] bsd.mp         [X] man60.tgz      [X] xshare60.tgz
Set name(s)? (or 'abort' or 'done') [done] -game* -x*
    [X] bsd            [X] base60.tgz     [ ] game60.tgz     [ ] xfont60.tgz
    [X] bsd.rd         [X] comp60.tgz     [ ] xbase60.tgz    [ ] xserv60.tgz
    [X] bsd.mp         [X] man60.tgz      [ ] xshare60.tgz
Set name(s)? (or 'abort' or 'done') [done]
Directory does not contain SHA256.sig. Continue without verification? [no] yes_
```

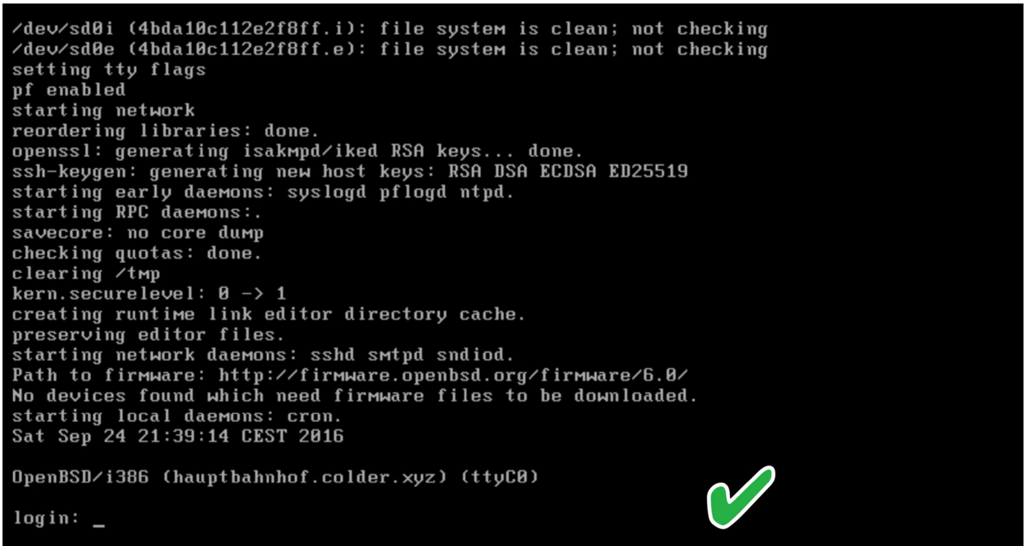
*Επιλογή σετ αρχείων που *δεν* θέλουμε να εγκατασταθούν. Για το firewall/router δεν χρειάζονται τα σετ περί παιχνιδιών ή παραθυρικού συστήματος X. Παρατηρήστε ότι για την αποεπιλογή δίνουμε μοτίβα που χωρίζονται με ένα κενό κι έχουν ως επίθεμα την παύλα. Προσέξτε εξάλλου και την τελευταία ερώτηση, για τις υπογραφές SHA256 που λείπουν. Αυτό είναι φυσιολογικό, πράγματι δεν περιλαμβάνονται στο δωρεάν ISO image που κατεβάσαμε, οπότε πληκτρολογήστε "yes" (χωρίς τα εισαγωγικά) και πιέστε το [Enter] για να ξεκινήσει η εγκατάσταση.*

```
Set name(s)? (or 'abort' or 'done') [done] -game* -x*
    [X] bsd            [X] base60.tgz     [ ] game60.tgz     [ ] xfont60.tgz
    [X] bsd.rd         [X] comp60.tgz     [ ] xbase60.tgz    [ ] xserv60.tgz
    [X] bsd.mp         [X] man60.tgz      [ ] xshare60.tgz
Set name(s)? (or 'abort' or 'done') [done]
Directory does not contain SHA256.sig. Continue without verification? [no] yes
Installing bsd         100% |***************************| 10622 KB   00:00
Installing bsd.rd      100% |***************************|  7005 KB   00:00
Installing bsd.mp      100% |***************************| 10661 KB   00:00
Installing base60.tgz  100% |***************************| 51113 KB   00:07
Extracting etc.tgz     100% |***************************|   186 KB   00:00
Installing comp60.tgz  100% |***************************| 44843 KB   00:06
Installing man60.tgz   100% |***************************|  8617 KB   00:01
Location of sets? (cd0 disk http or 'done') [done]
Time appears wrong.  Set to 'Sat Sep 24 21:34:14 CEST 2016'? [yes]
Saving configuration files...done.
Making all device nodes...done.
Multiprocessor machine; using bsd.mp instead of bsd.

CONGRATULATIONS! Your OpenBSD install has been successfully completed!
To boot the new system, enter 'reboot' at the command prompt.
When you login to your new system the first time, please read your mail
using the 'mail' command.

# reboot_
```

*Πολύ σύντομα η εγκατάσταση των σετ ολοκληρώνεται, δεχόμεθα συγχαρητήρια και καλούμαστε να πληκτρολογήσουμε "reboot" (χωρίς τα εισαγωγικά), προκειμένου ο υπολογιστής να εκκινήσει για πρώτη φορά στο ωραίο OpenBSD.*




*Πρώτο boot στο ολοκαίνουργιο OpenBSD μας, το οποίο θα αποτελέσει τη βάση για το ασφαλέστατο firewall/router του τοπικού μας δικτύου. Παρεμπιπτόντως, πώς σας φαίνεται το FQDN του συστήματος;*

## Εύκολες κι ασφαλείς συνδέσεις, έλεγχος δικτύωσης

Πρώτο μέλημά μας μετά την εγκατάσταση του OpenBSD είναι τα passwordless SSH logins από άλλο μηχάνημα του τοπικού δικτύου. Για το τι ακριβώς εννοούμε αλλά και για το πώς επιτυγχάνονται τα προαναφερθέντα logins, διαβάστε το σχετικό άρθρο στο site του περιοδικού (https://deltahacker.gr/?p=13357). Η δικτύωση του OpenBSD box λογικά πρέπει να 'ναι σωστά ρυθμισμένη, κι αυτό μπορούμε να το διαπιστώσουμε, π.χ., με ένα ping στο google.com. Προαιρετικά, αξίζει να ρίξουμε μια ματιά σε ορισμένα αρχεία όπου βρίσκονται παράμετροι κι επιλογές περί δικτύωσης. Δείτε το screenshot που ακολουθεί και διαβάστε τη συνοδευτική περιγραφή.

Πριν συνεχίσουμε οφείλουμε να ενεργοποιήσουμε το IP forwarding για τον router μας, ώστε να επιτρέπεται η προώθηση δικτυακών πακέτων μεταξύ network interfaces. Αρκεί να πληκτρολογήσουμε

```
# echo "net.inet.ip.forwarding=1" > /etc/sysctl.conf
```

και το IP forwarding θα ενεργοποιείται σε κάθε reboot. Δεν είναι ενεργοποιημένο τώρα, όμως σε λίγο που θα θέλουμε να δούμε αν όλα είναι εντάξει, θα κάνουμε και μια επανεκκίνηση.

```
# cat /etc/hostname.em0
inet 192.168.201.250 255.255.255.0
#
# cat /etc/hostname.em1
inet 172.16.164.250 255.255.255.0
#
# cat /etc/mygate
192.168.201.2
#
# cat /etc/resolv.conf
lookup file bind
nameserver 192.168.201.2
#
# cat /etc/hosts
127.0.0.1          localhost
::1                localhost
192.168.201.250 hauptbahnhof.colder.xyz hauptbahnhof
172.16.164.250  husavik.colder.xyz husavik
#
# cat /etc/myname
hauptbahnhof.colder.xyz
# _
```

*Οι επιλογές δικτύωσης που κάναμε κατά το στάδιο της εγκατάστασης του OpenBSD, βρίσκονται όλες αποθηκευμένες στα αρχεία απλού κειμένου hostname.em0, hostname.em1, mygate, resolv.conf, hosts και myname – όλα εντός του καταλόγου /etc. Αναλυτικότερα, στα hostname.em0 και hostname.em1 βρίσκουμε τη διεύθυνση IP και το netmask καθενός εκ των em0 και em1 interfaces αντίστοιχα [(1) και (2)]. Στο δε mygate είναι η διεύθυνση της πύλης, δηλαδή του LAN interface που έχει το modem/router του ISP μας (3). Στο resolv.conf μεταξύ άλλων βλέπουμε τη διεύθυνση IP του nameserver που προσωρινά χρησιμοποιεί το OpenBSD [(4), είναι ίδια με τη διεύθυνση της πύλης]. Στο hosts υπάρχουν αντιστοιχίσεις μεταξύ των IP των em0, em1, FQDNs και hostnames (5). Τέλος, στο αρχείο myname είναι το FQDN του WAN interface, δηλαδή του em0 (6).*

## Υπηρεσίες DHCP και DNS, με το dnsmasq

Η υπηρεσία DHCP είναι σχεδόν πάντα απαραίτητη στους routers, αφού όλες οι συσκευές-πελάτες καλό είναι να 'χουν δυνατότητα για αυτόματη διευθυνσιοδότηση. Πιθανώς βέβαια να υπάρχουν και κάποιες εξ αυτών για τις οποίες θα θέλουμε στατικά IP. Ακόμη και τότε όμως καλό είναι να μην τους ορίζουμε τα IP χειροκίνητα, αλλά να τους τα αντιστοιχίζει ο DHCP server βάσει MAC address. Επίσης, σίγουρα οι συσκευές μας χρειάζονται όλες έναν nameserver, ο οποίος θα φροντίζει και θα μεταφράζει τα domain names σε IPs. Μετά την εγκατάσταση, το OpenBSD διαθέτει τόσο DHCP όσο και DNS server (τον unbound). Το θέμα είναι ότι *δεν* συνεργάζονται

για το resolving των FQDNs που έχουν τα μηχανήματα του τοπικού δικτύου. Παράδειγμα: Αν ένα μηχάνημα έχει FQDN το einarben.colder.xyz, δεν μπορούμε από άλλο μηχάνημα να κάνουμε, π.χ., ping einarben.colder.xyz. Στη θέση του einarben.colder.xyz θα πρέπει να βάλουμε την αριθμητική διεύθυνση IP που έχει.

Ευτυχώς, αντί να χρησιμοποιήσουμε τους προκαθορισμένους DHCP και DNS servers, κάλλιστα μπορούμε να εγκαταστήσουμε τον dnsmasq: παρέχει υπηρεσίες DHCP αλλά *και* DNS, ενώ κάνει και resolving ονομάτων που έχουν αντιστοιχιστεί από τον DHCP. Ο dnsmasq δεν υπάρχει στη βασική εγκατάσταση του OpenBSD, οπότε θα καταφύγουμε στο εργαλείο pkg_add ώστε να φέρουμε το σχετικό πακέτο από το Internet και να το εγκαταστήσουμε. Με την ευκαιρία, στο αρχείο /root/.profile ας προσθέσουμε μια γραμμή που θα υποδεικνύει από πού θα κατεβαίνουν τα πακέτα:

```
# echo "export PKG_PATH=http://openbsd.cs.fau.de/pub/OpenBSD/6.0/packages/i386"
>> .profile
```

Οι παρατηρητικοί θα προσέξατε ότι ξανά προτιμήσαμε τον mirror στο Erlangen. Παρεμπιπτόντως, ας σημειώσουμε ότι μια λίστα με τα διαθέσιμα mirrors θα βρείτε στο

https://www.openbsd.org/ftp.html

δηλαδή στη σελίδα download που επισκεφθήκαμε προηγουμένως, για τη λήψη του OpenBSD. Σε κάθε περίπτωση, κατά τον ορισμό package mirror προσέξτε την έκδοση του λειτουργικού (για εμάς 6.0), καθώς και την αρχιτεκτονική που έχετε εγκαταστήσει (για εμάς i386). Προκειμένου να ληφθεί άμεσα υπόψη η τιμή της μεταβλητής PKG_PATH, στο τερματικό δίνουμε:

```
# export PKG_PATH=http://openbsd.cs.fau.de/pub/OpenBSD/6.0/packages/i386
```

Για την εγκατάσταση του πακέτου dnsmasq, γράφουμε:

```
# pkg_add dnsmasq
quirks-2.241 signed on 2016-07-29T15:39:09Z
quirks-2.241: ok
dnsmasq-2.76: ok
The following new rcscripts were installed: /etc/rc.d/dnsmasq
See rcctl(8) for details.
```

Τέλεια. Τώρα, το αρχείο ρυθμίσεων που θέλουμε είναι το /etc/dnsmasq.conf και πριν το πειράξουμε ας κρατήσουμε ένα αντίγραφό του:

```
# cp /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
```

Για την τροποποίηση του /etc/dnsmasq.conf υπάρχει το vi. Αν ο συγκεκριμένος editor δεν είναι του γούστου σας, εγκαταστήστε το nano με το εργαλείο pkg_add.

```
# vi /etc/dnsmasq.conf
```

Προτείνουμε να σβήσετε όλες τις γραμμές του αρχείου ρυθμίσεων και να ξεκινήσετε με ένα απλό setup. Δείτε, π.χ., τα περιεχόμενα που έχει το /etc/dnsmasq.conf στον δικό μας OpenBSD router:

```
no-resolv
proxy-dnssec
server=127.0.0.1#40
listen-address=127.0.0.1,172.16.164.250
bind-interfaces

domain=colder.xyz
dhcp-range=172.16.164.100,172.16.164.125,12h

dhcp-host=00:0c:29:ec:27:8e,einarben,172.16.164.195,24h
dhcp-host=00:0c:29:71:71:b8,thesender,172.16.164.200,24h
dhcp-host=00:50:56:2b:5b:c8,ohsuse,172.16.164.205,24h
dhcp-host=00:50:56:2f:e5:78,osuse-vm,172.16.164.210,24h
dhcp-host=00:0c:29:cb:fa:4a,tweed,172.16.164.215,24h

dhcp-option=252,"\n"
```

Προσέξτε την οδηγία server, στην τρίτη γραμμή από πάνω, η οποία αφορά στο DNS-μέρος του dnsmasq: αν για το ερώτημα κάποιου πελάτη δεν υπάρχει απάντηση στην cache του dnsmasq, τότε ρωτά κάποια υπηρεσία που τρέχει τοπικά κι αφουγκράζεται για αιτήσεις από το port 40. Πρόκειται για το DNSCrypt, το οποίο δεν έχουμε εγκαταστήσει ακόμα αλλά θα το φροντίσουμε σε λίγο. Προσοχή χρειάζεται και η οδηγία listen-address, από κάτω. Όπως βλέπετε, το dnsmasq εξυπηρετεί πελάτες από το ίδιο το μηχάνημα (127.0.0.1), καθώς και πελάτες από το τοπικό δίκτυο (θυμηθείτε: το LAN interface στο μηχάνημά μας έχει για IP το 172.16.164.250). Κατά τα άλλα, οι διευθύνσεις IP που μοιράζει αυτόματα το DHCP-μέρος του dnsmasq, βρίσκονται εντός του διαστήματος 172.16.164.100-192.16.164.125 (26 στο πλήθος διαφορετικές συσκευές δεν είναι και λίγες για ένα οικιακό δίκτυο). Επίσης, πέντε συγκεκριμένοι πελάτες, δηλαδή πέντε μηχανήματα με συγκεκριμένα MAC addresses, παίρνουν πάντα την ίδια διεύθυνση IP *εκτός* του διαστήματος εκείνων που μοι-

ράζονται δυναμικά. Με βάση τα περιεχόμενα του dnsmasq.conf που μόλις παραθέσαμε, τροποποιήστε αναλόγως το δικό σας. Διαβάστε αν θέλετε και τα σχόλια στο πρωτότυπο configuration file, δηλαδή στο dnsmasq.conf.orig.

Αφού τελειώσουμε με τις αλλαγές και τις αποθηκεύσουμε, οφείλουμε να πράξουμε τα δέοντα ώστε η υπηρεσία να ξεκινά αυτόματα κατά την εκκίνηση του OpenBSD. Προς τούτο, δημιουργούμε το νέο αρχείο /etc/rc.conf.local και του προσθέτουμε μία μόνο γραμμή. Δείτε:

```
# echo 'pkg_scripts="dnsmasq"' > /etc/rc.conf.local
```

(προσοχή στα απλά και στα διπλά εισαγωγικά). Δεν θα ενεργοποιήσουμε ακόμα το dnsmasq. Έχουμε να φροντίσουμε για το DNSCrypt και φυσικά να ορίσουμε τους κατάλληλους κανόνες στο firewall.

## Ενίσχυση privacy με το DNSCrypt

Το DNSCrypt (https://dnscrypt.org) είναι ένα πρωτόκολλο για την ταυτοποίηση των συνδέσεων μεταξύ DNS client και DNS server, καθιστώντας αδύνατες τις επιθέσεις DNS spoofing. Με χρήση κρυπτογραφίας και ψηφιακών υπογραφών αφενός διασφαλίζεται ότι οι απαντήσεις προέρχονται πράγματι από τον επιλεγμένο nameserver, αφετέρου ότι δεν έχουν τροποποιηθεί στην πορεία. Υλοποιήσεις πελατών που γνωρίζουν το DNSCrypt διατίθενται για διάφορα λειτουργικά συστήματα, του OpenBSD συμπεριλαμβανομένου. Κατά τα αναμενόμενα, η εγκατάσταση του σχετικού πακέτου γίνεται με το εργαλείο pkg_add:

```
# pkg_add dnscrypt-proxy
quirks-2.241 signed on 2016-07-29T15:39:09Z
dnscrypt-proxy-1.6.1p0:libltdl-2.4.2p1: ok
dnscrypt-proxy-1.6.1p0:libsodium-1.0.10: ok
dnscrypt-proxy-1.6.1p0:libexecinfo-0.3v0: ok
dnscrypt-proxy-1.6.1p0: ok
The following new rcscripts were installed: /etc/rc.d/dnscrypt_proxy
See rcctl(8) for details.
Look in /usr/local/share/doc/pkg-readmes for extra documentation.
```

Ανοίγουμε στη συνέχεια το /etc/rc.conf.local με τον text editor της προτίμησής μας

```
# vi /etc/rc.conf.local
```

και τροποποιούμε τα περιεχόμενα του αρχείου ώστε να είναι όπως τα ακόλουθα:

```
dnscrypt_proxy_flags=-l /dev/null -R ns0.dnscrypt.is -a 127.0.0.1:40
pkg_scripts="dnsmasq dnscrypt_proxy"
```

Αποθηκεύουμε τις αλλαγές και προς το παρόν δεν ενεργοποιούμε το DNSCrypt. Όπως είναι φανερό από την πρώτη γραμμή του rc.conf.local, ο απομακρυσμένος DNSCrypt-aware nameserver που υποδείξαμε στο DNSCrypt είναι ο ns0.dnscrypt.is (εις το Reykjavik). Εννοείται ότι υπάρχει δυνατότητα καθορισμού διαφορετικού nameserver και μια λίστα με DNSCrypt resolvers θα βρείτε στο ακόλουθο URL:

https://github.com/jedisct1/dnscrypt-proxy/blob/master/dnscrypt-resolvers.csv

Παρατηρήστε, εξάλλου, ότι ο DNSCrypt ακούει για συνδέσεις από το localhost και συγκεκριμένα από το port 40. Πολύ ευχάριστη αυτή η σύμπτωση, ειδικά αν σκεφτεί κανείς ότι εκεί είπαμε στον dnsmasq να στέλνει τα αιτήματά του όταν δεν μπορεί να εξυπηρετήσει πελάτες κοιτάζοντας στην τοπική DNS cache. Ας φροντίσουμε ώστε και το ίδιο το λειτουργικό να χρησιμοποιεί τον dnsmasq. Απλά ανοίγουμε το αρχείο /etc/resolv.conf προς επεξεργασία και φροντίζουμε ώστε τα περιεχόμενά του να μοιάζουν με τα ακόλουθα:

```
search colder.xyz
nameserver 127.0.0.1
lookup file bind
```

Για τα DNS lookups, η τελευταία γραμμή λέει ότι *πρώτα* πρέπει να ελέγχεται το αρχείο /etc/hosts και *μετά* ο nameserver. Και ποιος είν' αυτός; Μα, αυτός που αναφέρεται στη γραμμή από πάνω και χρησιμοποιεί το προκαθορισμένο port (το 53), δηλαδή ο dnsmasq. (Και ναι, πρακτικά ακούσαμε τι είπατε κάποιοι όταν διαβάσατε το "και ποιος είν' αυτός".)

## Ρύθμιση firewall

Ωραία, έως εδώ έχουμε DHCP και DNS (dnsmasq), ενώ το resolving των διευθύνσεων που δεν βρίσκονται στην cache του nameserver επιτυγχάνεται μέσω του πρωτοκόλλου DNSCrypt. Πρέπει τώρα να ρυθμίσουμε και το firewall του router μας. Ουσιαστικά θα τροποποιήσουμε το αρχείο /etc/pf.conf. Δείτε τα περιεχόμενα του δικού μας:

```
theLAN="em1"

table <notReally> { 0.0.0.0/8 10.0.0.0/8 127.0.0.0/8 169.254.0.0/16 \
                     172.16.0.0/12 192.0.0.0/24 192.0.2.0/24 224.0.0.0/3 \
                     192.168.0.0/16 198.18.0.0/15 198.51.100.0/24 \
                     203.0.113.0/24 }
```

```
set block-policy drop
set loginterface egress
set skip on lo0

match in all scrub (no-df random-id max-mss 1440)
match out on egress inet from !(egress:network) to any nat-to (egress:0)

antispoof quick for (egress)
block in quick on egress from <notReally> to any
block in quick inet6 all
block return out quick inet6 all
block return out quick log on egress from any to <notReally>
block return out quick log on egress proto { tcp udp } from any to any port 53
block in all

pass out quick inet keep state
pass in on $theLAN inet
pass in on egress inet proto tcp from any to (egress) port 22
pass in on $theLAN inet proto { tcp udp } from $theLAN to ! $theLAN port 53
rdr-to $theLAN
```

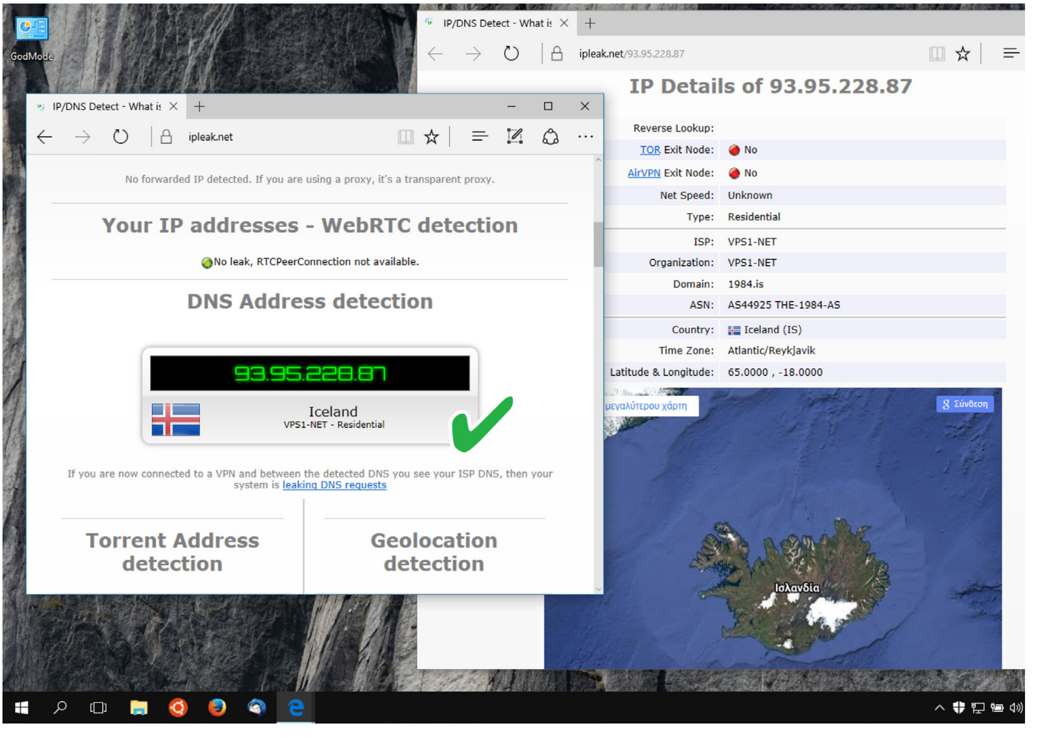
Η περιγραφή των κανόνων του pf ξεφεύγει από τους σκοπούς του παρόντος. Το καλό πάντως με το συντακτικό του pf είναι ότι επιτρέπει τη δημιουργία κανόνων οι οποίοι "διαβάζονται" εύκολα. Μπορείτε βεβαίως να συμβουλευτείτε και προηγούμενα άρθρα μας, όπως, π.χ., αυτό που ξεκινά από τη σελίδα 74 του τεύχους 022 (https://deltahacker.gr/deltahacker022) ή εκείνο που ξεκινά από τη σελίδα 62 του τεύχους 023 (https://deltahacker.gr/deltahacker023). Εναλλακτικά, μελετήστε το επίσημο PF User's Guide (https://www.openbsd.org/faq/pf/index.html) και ειδικά την ενότητα Example rulesets / Building a router (https://www.openbsd.org/faq/pf/example1.html). Πριν οτιδήποτε άλλο, αξίζει στο σημείο αυτό να κάνουμε μια επανεκκίνηση του συστήματς:

```
# reboot
```

Λογικά, μετά κι απ' αυτό, όλες οι υπηρεσίες που παρέχει το firewall/router θα πρέπει να είναι ενεργές κι εκείνο πανέτοιμο για να εξυπηρετεί πελάτες *πίσω* από το LAN interface.

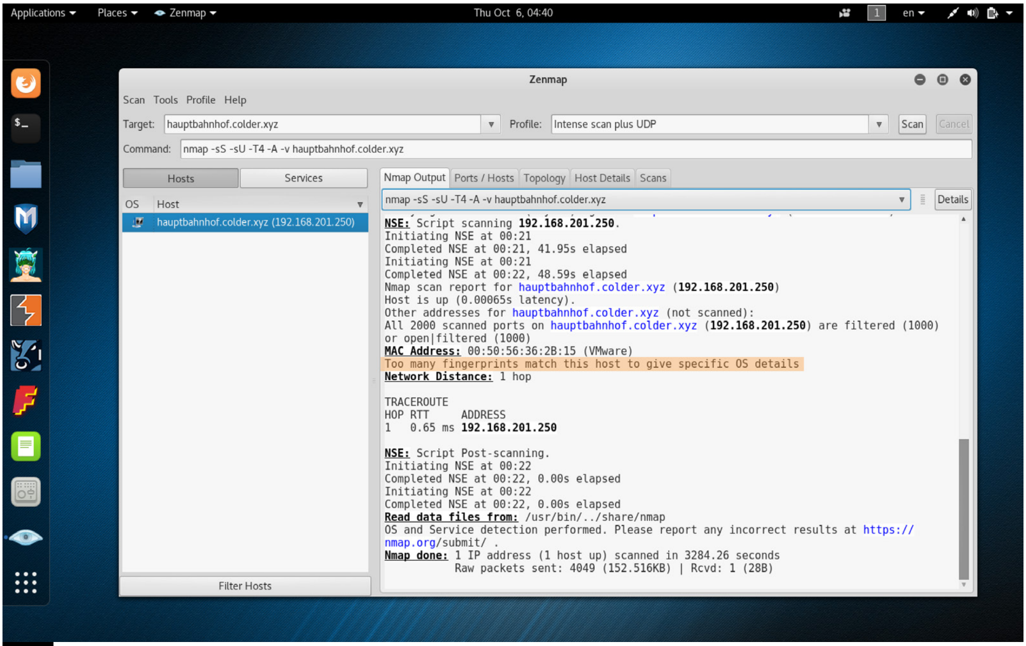
## Δοκιμές

Αυτή τη στιγμή έχουμε ένα ελαφρύ, ισχυρό κι εξαιρετικά ασφαλές OpenBSD-based firewall/router για το LAN μας. Οι πρώτες δοκιμές που οφείλουμε να κάνουμε ώστε να βεβαιωθούμε ότι δουλεύει, είναι να χρησιμοποιήσουμε ένα οποιοδήποτε μηχάνημα *πίσω* από τον ολοκαίνουργιο router, για πρόσβαση στο τοπικό δίκτυο και βεβαίως στο Internet. Από εκεί και πέρα, μια άλλη δοκιμή που δεν πρέπει να παραλείψουμε, αφορά στη χρήση του DNSCrypt. Στην περίπτωσή μας, για παράδειγμα, σε ένα μηχάνημα με Windows 10 που εξυπηρετούνταν από το firewall/router, ανοίξαμε τον Edge browser κι επισκεφτήκαμε τη διεύθυνση https://ipleak.net. Μετά από λίγα δευτερόλεπτα διαπιστώσαμε ότι ο DNS server που μας εξυπηρετούσε ήταν στην Ισλανδία – και το γεγονός αυτό είναι συνεπές με την οδηγία που έχουμε στο αρχείο /etc/rc.conf.local.




*Με μια επίσκεψη στο http://ipleak.net διαπιστώνουμε ότι ο nameserver που χρησιμοποιούμε είναι στην Ισλανδία. Ακριβώς αυτό έχουμε ζητήσει με τη σχετική οδηγία στο /etc/rc.conf.local, η οποία λαμβάνεται υπόψη πριν από την ενεργοποίηση της υπηρεσίας του DNSCrypt.*

Τέλος, αξίζει να σκανάρουμε το WAN interface του firewall/router από έναν υπολογιστή που βρίσκεται πίσω από τον router του ISP. Δείτε το screenshot που ακολουθεί.




*Σκανάρισμα του WAN interface του router μας από μηχάνημα με Kali Linux, το οποίο βρίσκεται εκτός του LAN που εξυπηρετεί ο router. Χρησιμοποιούμε το nmap μέσα από το front-end περιβάλλον που παρέχει το zenmap, και μόλις ολοκληρώσαμε μια σάρωση με βάση το προφίλ Intense scan plus UDP. Όπως διαπιστώνετε, ο router δεν αποκαλύπτει και πολλά για τον εαυτό του.*

## Βελτιώσεις

Θα συμφωνήσετε φανταζόμαστε ότι ένας router δεν χρειάζεται υποσύστημα ήχου. Καλό είναι όμως να κάνει εξοικονόμηση ενέργειας – αν κι όποτε μπορεί φυσικά. Η απενεργοποίηση του sound daemon και η ενεργοποίηση του power management daemon γίνονται με τις κατάλληλες προσθήκες στο αρχείο /etc/rc.conf.local. Δείτε το πλήρες περιεχόμενο του δικού μας:

```
sndiod_flags=NO
apmd_flags="-A"

dnscrypt_proxy_flags=-l /dev/null -R ns0.dnscrypt.is -a 127.0.0.1:40
pkg_scripts="dnsmasq dnscrypt_proxy"
```

Με τις δύο πρώτες γραμμές διασφαλίζουμε ότι κατά την εκκίνηση του OpenBSD *δεν* θα ξεκινά ο sound daemon, θα ενεργοποιείται όμως η υπηρεσία εξοικονόμησης ενέργειας. Προκειμένου να ληφθούν άμεσα υπόψη οι δύο νέες οδηγίες στο /etc/rc.conf.local, χωρίς να επανεκκινήσουμε το λειτουργικό, γράφουμε:

```
# /etc/rc.d/sndiod stop
# /etc/rc.d/apmd start
```

Τέλος, στην περίπτωση που έχετε εγκαταστήσει το OpenBSD σε flash drive καλό είναι να ελαχιστοποιήσετε τις εγγραφές επί αυτού. Ανοίξτε το αρχείο /etc/fstab με τον editor της προτίμησής σας

```
# vi /etc/fstab
```

και τροποποιήστε τη γραμμή με τις οδηγίες προσάρτησης του root partition. Για παράδειγμα, η σχετική γραμμή στο δικό μας fstab είχε ως εξής

```
4bda10c112e2f8ff.a / ffs rw 1 1
```

και την αλλάξαμε σε

```
4bda10c112e2f8ff.a / ffs rw,noatime,softdep 1 1
```

Οι νέες παράμετροι για την προσάρτηση λαμβάνονται υπόψη από την επόμενη επανεκκίνηση του λειτουργικού συστήματος.


CMYK & WEB
cmykandweb.gr
GRAPHIC DESIGN
design • layout • web

**Skill:** Intermediate
**Tags:** SQLIte, SQL, databases, Python

# SQLite
# Μια Βάση και στα Γρήγορα
## [Μέρος 2/2]

**Στο πρώτο μέρος της μίνι σειράς μας ασχοληθήκαμε με το πιο θεωρητικό τμήμα SQLite, της περίφημης αυτής RDBMS/βιβλιοθήκης. Στο παρόν άρθρο, η λογική σειρά των πραγμάτων μάς καλεί να εφαρμόσουμε όσα μάθαμε μέχρι στιγμής στην πράξη.**

*του Πέτρου Κυλαδίτη*

Όπως ήδη είπαμε στο πρώτο μέρος της μίνι σειράς μας (https://deltahacker.gr/actsubs-sqlite-p1), κώδικα της SQLite μπορούμε να ενσωματώνουμε στις εφαρμογές μας. Κι αν δεν θέλουμε να ανακατευόμαστε με τη C επιλέγουμε κάποιον από τους wrappers που κυκλοφορούν, φέρνοντας έτσι τη λειτουργικότητα και τις δυνατότητες της SQLite στον εγγενή τρόπο διαχείρισης βάσεων δεδομένων της αγαπημένης μας γλώσσας. Σε μια προσπάθεια να ξεφύγουμε από τα παραδείγματα υλοποίησης web εφαρμογών και να σας αποδείξουμε ότι η SQLite δεν περιορίζεται μόνο σ' αυτά τα περιβάλλοντα, αλλά κάνει και για την ανάπτυξη desktop προγραμμάτων, θα γράψουμε σε Python μια μικρή εφαρμογή διαχείρισης των όποιων domains έχουμε στην κατοχή μας. Στην εν λόγω εφαρμογή, η οποία θα λειτουργεί σε περιβάλλον κονσόλας κειμένου, θα εισάγουμε τα στοιχεία των domains μας: όνομα domain, καταχωρητής (registrar), name servers που το εξυπηρετούν, ημερομηνία λήξης και κόστος.

Ας ξεκινήσουμε από τη βάση. Θα χρειαστεί να κατασκευάσουμε έναν πίνακα με την ακόλουθη δομή:

```
==================================================
Πίνακας domains
==================================================
Πεδίο    |  Τύπος
--------------------------------------------------
id       |  INTEGER <-- primary key
domain   |  TEXT
registrar|  TEXT
dns1     |  TEXT
dns2     |  TEXT
renew_date  |  TEXT
cost     |  FLOAT
--------------------------------------------------
```

Για την υλοποίηση της βάσης μπορούμε να καταφύγουμε σε πάμπολλες μεθόδους, δύο από τις οποίες είδαμε στο πρώτο μέρος της μίνι σειράς μας (https://deltahacker.gr/actsubs-sqlite-p1). Η πρώτη είναι με τη χρήση του εργαλείου κονσόλας slqite3, η δεύτερη με τη χρήση του εργαλείου web UI ονόματι phpLiteAdmin. Επιδεικτικά κι ευθαρσώς αγνοούμε αμφότερες τις μεθόδους και φροντίζουμε ώστε η δημιουργία του πίνακα να γίνεται μέσα από το ίδιο το πρόγραμμά μας! Από τη μια επιθυμούμε να εξασκηθούμε στη χρήση της SQLite μέσω Python, από την άλλη θέλουμε η εφαρμογή μας να έχει μια εσάνς "αυτοΐασης". Πιο συγκεκριμένα, όταν για οποιονδήποτε λόγο δεν βρίσκεται η βάση, αντί η εφρμογή μας να κρεμάει θα την αναδημιουργεί. Ίσως τώρα αναρωτιέστε πόσο δύσκολο είναι να υλοποιηθεί κάτι τέτοιο. Σας πληροφορούμε ότι δεν είναι καθόλου δύσκολο. Και για του λόγου το αληθές, ας πάμε να εξετάσουμε αποσπασματικά τον κώδικά μας.

```
import sqlite3
import os.path
import datetime
from dateutil.relativedelta import relativedelta

DB_NAME = "domains.db"
titles = ("Domain","Registrar","Primary DNS","Secondary DNS","Renewal Date","Cost")
max_titles_len = len(max(titles, key=len))
```

Για αρχή εισάγουμε τις απαραίτητες βιβλιοθήκες, στοιχεία των οποίων θα αξιοποιήσουμε στο συγκεκριμένο project μας. Η πρώτη απ' αυτές είναι η sqlite3, η οποία δίνει πρόσβαση στα στοιχεία διαχείρισης του ομώνυμου συστήματος. Η δεύτερη είναι η os.path. Θα μας φανεί χρήσιμη για την υλοποίηση της λεγόμενης μεθόδου "αυτοΐασης". Η τρίτη βιβλιοθήκη θα μας χρησιμεύσει για τη διαχείριση των ημερομηνιών. Η τελευταία, με το ομώνυμο αντικείμενο, μας παρέχει πολύ μεγάλη ευελιξία στη διαχείριση των δεδομένων μεταξύ ημερομηνιών.

Κατόπιν της εισαγωγής των βιβλιοθηκών, αρχικοποιούμε κάποιες μεταβλητές. Η DB_NAME, την οποία θα χρησιμοποιήσουμε στη συνέχεια ως σταθερά, επιστρέφει τη διαδρομή προς το αρχείο που αποτελεί τη βάση δεδομένων. Αν δεν ορίσουμε μονοπάτι, αυτό νοείται ως ο τρέχων φάκελος μέσα στον οποίο βρίσκεται το πρόγραμμά μας. Στην επόμενη γραμμή, σε ένα tuple ορίζουμε μια σειρά λεκτικών που θα χρησιμοποιήσουμε σε διάφορες από τις μεθόδους μας. Όπως ίσως θα διαπιστώσατε ήδη, αφορούν στους τίτλους των πεδίων του μοναδικού πίνακα της βάσης μας. Στην τελευταία γραμμή του αποσπάσματος, χρησιμοποιώντας τη μέθοδο max εντοπίζουμε το μεγαλύτερο σε μήκος μέλος του tuple – και σε συνδυασμό με τη συνάρτηση len βρίσκουμε το μήκος αυτό. Ίσως εύλογα να απορείτε για το λόγο που μπαίνουμε σε μια τέτοια διαδικασία. Η αλήθεια είναι πως επιθυμούμε να έχουμε ένα όμορφο οπτικά αποτέλεσμα στην κονσόλα μας, οπότε το μέγεθος αυτό θα μας βοηθήσει να στοιχίσουμε σωστά τα στοιχεία στην οθόνη.

```
009
010 def init():
011     if not os.path.exists(DB_NAME):
012         conn = sqlite3.connect(DB_NAME)
013         curs = conn.cursor()
014         curs.execute("CREATE TABLE domains(id INTEGER PRIMARY KEY, domain
            TEXT, registrar TEXT, dns1 TEXT, dns2 TEXT, renew_date TEXT, cost
            FLOAT)")
015         curs.close()
016     print "myDomains management system - A pythonic sqlite example\
        nCopyright (c) 2016, Petros Kyladitis <multipetros.gr>\n\n"
017     mainloop()
018
```

Στη γραμμή 10 ξεκινά η αρχική συνάρτηση της εφαρμογής μας, η οποία είναι και υπεύθυνη για την αναδημιουργία της βάσης δεδομένων σε περίπτωση που αυτή δεν εντοπιστεί. Στην γραμμή 11 γίνεται ο σχετικός έλεγχος ύπαρξης του αρχείου της βάσης. Αν δεν βρεθεί, δημιουργούμε ένα αντικείμενο Connection της sqlite παρέχοντάς του τη σχετική διαδρομή της βάσης. Απ' αυτό στη συνέχεια λαμβάνουμε ένα αντικείμενο Cursor, μέσω του οποίου θα χειριστούμε τη βάση μας εκτελώντας

εντολές SQL και λαμβάνοντας τα αποτελέσματα της εκτέλεσής τους. Στη γραμμή 014 βλέπουμε την εκτέλεση μιας εντολής SQL για τη δημιουργία του πίνακα της βάσης μας, με τη χρήση της μεθόδου execute. Στη συνέχεια κλείνουμε τη σύνδεση του αντικειμένου Cursor με τη βοήθεια της μεθόδου close(). Προς το τέλος τώρα της συνάρτησής μας –κι ανεξαρτήτως της δημιουργίας ή όχι της βάσης–, εμφανίζουμε ένα μήνυμα με πληροφορίες για την εφαρμογή και καλούμε τη συνάρτηση mainloop(), η οποία είναι το βασικό μενού του προγράμματός μας.

```
019
020 def add_domain():
021     print "Enter domain details. You can cancel the input by entering '.c'"
022     details = ""
023     details_data = []
024     for i in range(6):
025         while details == "":
026             details = raw_input( titles[i].ljust( max_titles_len+2, ".") + ": "
027         if details == ".c":
028             return
029         details_data.append(details)
030         details = ""
031     conn = sqlite3.connect(DB_NAME)
032     curs = conn.cursor()
033     curs.execute("INSERT INTO domains (id, domain, registrar, dns1, dns2, renew_date, cost) VALUES (null, ?, ?, ?, ?, ?, ?)", tuple(details_data))
034     conn.commit()
035     curs.close()
```

Ας περάσουμε τώρα στην add_domain(), το όνομα της οποίας μας προϊδεάζει ότι πρόκειται για τη συνάρτηση καταχώρισης ενός νέου domain στη βάση. Αρχικά εμφανίζουμε μια προτροπή προς τον χρήστη για την εισαγωγή των δεδομένων, η οποία μπορεί να διακοπεί δίνοντας το λεκτικό ".c" (χωρίς τα εισαγωγικά). Παλιομοδίτικος τρόπος χειρισμού, όμως στην περίπτωση του παραδείγματός μας λειτουργεί ικανοποιητικά. Στη γραμμή 22 έχουμε τη μεταβλητή details, την οποία αρχικοποιούμε εκχωρώντας της μια κενή συμβολοσειρά. Αυτή θα τη χρησιμοποιήσουμε προκειμένου να λαμβάνουμε την είσοδο του χρήστη, την οποία θα προσαρτούμε κάθε φορά στο τέλος του πίνακα details_data. Για καθένα λοιπόν από τα στοιχεία του πίνακα (έξι στο σύνολο), εκτελούμε το βρόχο που ξεκινά στη γραμμή 024 κι εμφανίζουμε μια σχετική προτροπή εισαγωγής δεδομένων, η οποία αποτελείται από τον τίτλο του εκάστο-

τε πεδίου στοιχισμένο σύμφωνα με το μήκος του μεγαλύτερου απ' αυτά, όπως το βρήκαμε στη συνάρτηση init() και το αποθηκεύσαμε στη μεταβλητή max_titles_len. Για να πραγματοποιηθεί η στοίχιση χρησιμοποιούμε τη συνάρτηση ljust(), η οποία δέχεται ως παράμετρο το ελάχιστο μήκος που θέλουμε να έχει η συμβολοσειρά στην οποία την εφαρμόζουμε, καθώς και μια προαιρετική μεταβλητή με τον χαρακτήρα που θέλουμε να συμπληρώσει από τα δεξιά τη συμβολοσειρά ώστε να φτάσει στο επιθυμητό μήκος. Αν δεν δώσουμε αυτή τη δεύτερη μεταβλητή, τότε από προεπιλογή η συμβολοσειρά μας θα συμπληρωθεί με κενά. Προσέξτε ότι την εισαγωγή των δεδομένων την κλείνουμε σε ένα βρόχο, ο οποίος εκτελείται όσο η μεταβλητή details είναι μια κενή συμβολοσειρά. Όπως ίσως ήδη θα έχετε καταλάβει, ο βρόχος αυτός μπαίνει ώστε να αποτρέψουμε το χρήστη από την εισαγωγή κενών δεδομένων. Εντός του βρόχου τώρα, στη γραμμή 027, πραγματοποιούμε έλεγχο για το αν η συμβολοσειρά που εισήγαγε ο χρήστης είναι η συμβολοσειρά ακύρωσης της εισαγωγής. Αν είναι, τότε η συνάρτηση επιστρέφει άμεσα και η ροή του προγράμματος συνεχίζεται στον κεντρικό βρόχο/μενού. Αν η εισαγωγή των δεδομένων ολοκληρωθεί επιτυχώς, τότε αναλαμβάνουν δράση τα αντικείμενα της βιβλιοθήκης sqlite3. Όπως και στην init(), δημιουργούμε ένα Connection στο οποίο συνδέουμε το αρχείο της βάσης μας. Εξ' αυτού λαμβάνουμε ένα Cursor, με τη βοήθεια του οποίου στη γραμμή 033 εκτελούμε ένα ερώτημα εισαγωγής στον πίνακα domains της βάσης. Αξίζει να προσέξετε ότι το ερώτημα που εκτελούμε δεν το δημιουργούμε με την κλασική μέθοδο της συνένωσης γραμματοσειράς, αλλά με τη χρήση παραμέτρων. Συγκεκριμένα, τα αγγλικά ερωτηματικά αντικαθίστανται από τη μέθοδο, με τις τιμές του tuple, με τη σειρά που υπάρχουν σ' αυτό. Κάτι τέτοιο μας εξυπηρετεί στην περίπτωση που η σειρά των δεδομένων του tuple είναι ίδια με αυτή των παραμέτρων που θέλουμε να περάσουμε στο ερώτημα. Στην περίπτωση που αυτή ήταν μπερδεμένη, μπορούσαμε πάλι να χρησιμοποιήσουμε παραμέτρους αλλά ονομαστικές αυτή τη φορά, χρησιμοποιώντας αντί του tuple ένα λεξικό. Παράδειγμα:

```
curs.execute("INSERT INTO domains (id, domain, registrar, dns1, dns2, renew_
date, cost) VALUES (null, :id, :domain, :registrar, :dns1, :dns2, :renew_date,
:cost)", {"id": details_data[0], "domain": details_data[1], "registrar":
details_data[2], "dns1": details_data[3], "dns2": details_data[4], "renew_
date": details_data[5], "cost": details_data[6] })
```

Μετά την εκτέλεση του ερωτήματος, προσέξτε τη χρήση της συνάρτησης commit(). Αυτή είναι απαραίτητη σε ερωτήματα που τροποποιούν τα δεδομένα της βάσης (εκτός της δημιουργίας πινάκων), ώστε αυτά να γραφτούν στο αρχείο. Αν καλέσουμε τη συνάρτηση close(), προτού κάνουμε commit, παρότι το ερώτημά μας θα έχει εκτελεστεί επιτυχώς τα δεδομένα δεν θα γραφτούν στο αρχείο. Ίσως θυμάστε από το προηγούμενο άρθρο μας ότι η SQLite υποστηρίζει συναλλαγές, εξ ου και η αναγκαιότητα της χρήσης του commit.

```
036
037 def del_domain():
038     domain = ""
039     while domain == "":
040         domain = raw_input("Domain to delete: ")
041     if domain == ".c":
042         return
043     conn = sqlite3.connect(DB_NAME)
044     curs = conn.cursor()
045     curs.execute("DELETE FROM domains WHERE domain='" + domain + "'")
046     if curs.rowcount > 0:
047         print "\nDomain '" + domain + "' deleted!"
048     else:
049         print "\n'" + domain + "' not found! Nothing deleted."
050     curs.close()
```

Συνεχίζοντας, στη συνάρτηση del_domain(), την οποία χρησιμοποιούμε για να διαγράψουμε μια καταχώριση από τη βάση μας, δε θα εντοπίσουμε σημαντικές διαφορές στη λογική μας σε σχέση με τη μέθοδο add_domain(). Εδώ, αντί να ζητούμε πλήρη στοιχεία, παίρνουμε από τον χρήστη το όνομα του domain και με βάση αυτό εκτελούμε ένα ερώτημα διαγραφής. Για την εκτέλεση του ερωτήματος δεν χρησιμοποιούμε παραμέτρους, αλλά συνθέτουμε απευθείας τη συμβολοσειρά του ερωτήματος. Τέλος, στη γραμμή 046 εκτελούμε έναν έλεγχο της ιδιότητας rowcount, του αντικειμένου Cursor. Η συγκεκριμένη ιδιότητα επιστρέφει το πλήθος των στοιχείων που επηρεάστηκαν από την εκτέλεση του ερωτήματος. Αν λοιπόν αυτή έχει τιμή μεγαλύτερη από 0, τότε εμφανίζουμε ένα μήνυμα επιτυχούς διαγραφής. Διαφορετικά, επειδή το πιθανότερο είναι να μην υπήρχε το domain, εμφανίζουμε ένα σχετικό μήνυμα.

```
051
052 def edit_domain():
053     conn = sqlite3.connect(DB_NAME)
054     curs = conn.cursor()
055     domain = ""
056     while domain == "":
057         domain = raw_input("Domain to edit: ")
```

```
058         if domain == ".c":
059             return
060         else:
061             curs.execute("SELECT id, domain, registrar, dns1, dns2, renew_
                date, cost FROM domains WHERE domain='" + domain + "'")
062             domain_data = curs.fetchone()
063             if domain_data is None :
064                 print "Domain '" + domain + "' not found!"
065                 domain = ""
066     details = ""
067     details_data = []
068     print "Enter new data, or leave it blank to keep the existing values"
069     for i in range(1,7):
070         details = raw_input(titles[i-1].ljust(max_titles_len+1) + "[" +
            str(domain_data[i]) + "]: ")
071         if details == "":
072             details = domain_data[i]
073         if details == ".c":
074             return
075         details_data.append(details)
076     details_data.append(domain_data[0])
077     curs.execute("UPDATE domains SET domain=?, registrar=?, dns1=?, dns2=?,
        renew_date=?, cost=? WHERE id=?", tuple(details_data))
078     conn.commit()
079     curs.close()
```

Η συνάρτηση edit_domain(), ρόλος της οποίας είναι η επεξεργασία μιας υπάρχουσας εγγραφής, έχει λίγο μεγαλύτερη έκταση από τις υπόλοιπες. Αυτό είναι φυσικό, αφού με μια προσεκτική ματιά μπορούμε να διαπιστώσουμε ότι γίνονται δύο αλληλεπιδράσεις με τη βάση. Αρχικά εκτελούμε ένα ερώτημα επιλογής, όπου το ζητούμενο αντικείμενο είναι τα στοιχεία για το domain που εισάγει ο χρήστης. Τα στοιχεία αυτά αποθηκεύονται σε έναν πίνακα με όνομα domains_data, αξιοποιώντας τη μέθοδο fetchone() από το αντικείμενο Cursor, η οποία επιστρέφει το πρώτο από τη λίστα των αποτελεσμάτων εκτέλεσης της εντολής select. Στη συνέχεια, στη γραμμή 069, ξεκινάμε έναν βρόχο προτρέποντας τον χρήστη να εισάγει τα νέα στοιχεία για το συγκεκριμένο domain. Αξιοποιώντας τα δεδομένα που αντλήσαμε γι' αυτό μαζί με τις προτροπές, εμφανίζουμε και τις τρέχουσες τιμές των στοιχείων. Σε αντίθεση

με τις άλλες συναρτήσεις, η εισαγωγή κενής συμβολοσειράς λαμβάνει την έννοια τις διατήρησης των υπαρχόντων δεδομένων. Τα δεδομένα αυτά, ένα προς ένα τοποθετούνται στην ουρά του πίνακα details_data με τη βοήθεια της συνάρτησης append. Παρατηρείστε ότι μετά το πέρας του βρόχου εισάγουμε στο τέλος του πίνακα ένα ακόμα στοιχείο, το id της συγκεκριμένης εγγραφής. Δεν είναι τυχαία η επιλογή της θέσης, καθώς αμέσως μετά, στη γραμμή 077 εκτελούμε το ερώτημα με τη βοήθεια των παραμέτρων και των ερωτηματικών, αξιοποιώντας ένα tuple το οποίο φτιάχνουμε από τον πίνακα details_data με τη βοήθεια της συνάρτησης μετατροπής tuple().

```
080
081 def search_domain():
082     domain = ""
083     while domain == "":
084         domain = raw_input("Domain to show: ")
085         print "\n"
086     if domain == ".c":
087         return
088     conn = sqlite3.connect(DB_NAME)
089     curs = conn.cursor()
090     curs.execute("SELECT domain, registrar, dns1, dns2, renew_date, cost
        FROM domains WHERE domain='" + domain + "'")
091     entry = curs.fetchone()
092     if entry is None :
093         print "Domain '" + domain + "' not found!"
094     else:
095         for i in range(6):
096             if i != 4:
097                 print titles[i].ljust(max_titles_len+2, ".") + ": " +
                    str(entry[i])
098             else:
099                 today = datetime.date.today()
100                 endday = datetime.datetime.strptime(entry[i], "%Y-%m-%d")
101                 rd = relativedelta(endday, today)
102                 time_left = " [%(years)d years, %(months)d months, %(days)d
```

```
                    days left]" % rd.__dict__
103                 print titles[i].ljust(max_titles_len+2, ".") + ": "  +
                    str(entry[i]) + time_left
104     curs.close()
```

Σειρά στις συναρτήσεις που εξετάζουμε έχει η search_domain(), μια συνάρτηση αναζήτησης κι εμφάνισης πληροφοριών ενός domain. Η αναζήτηση γίνεται με τον γνωστό πλέον τρόπο εκτέλεσης ερωτημάτων επιλογής και, στην περίπτωση που βρεθεί η ζητούμενη καταχώριση, εμφανίζονται ένα προς ένα τα στοιχεία του domain. Μια ιδιαιτερότητα αφορά στο στοιχείο της ημερομηνίας λήξης του domain, όπου προς διευκόλυνση των χρηστών της εφαρμογής μας υπολογίζουμε το χρονικό διάστημα που απομένει ως τη λήξη αυτού. Για τον υπολογισμό του χρησιμοποιούμε τη μεταβλητή today, στην οποία αποθηκεύουμε ένα αντικείμενο date με τη τρέχουσα ημερομηνία. Για τη μεταβλητή endday, η οποία αφορά στην ημερομηνία λήξης του domain, δημιουργούμε ένα αντικείμενο date από τη συμβολοσειρά που είναι αποθηκευμένη στη βάση, στη μορφή "ΕΕΕΕ-ΜΜ-ΗΗ" κι αξιοποιώντας τη συνάρτηση μετατροπής strptime(). Στη συνέχεια, στη γραμμή 101, δημιουργούμε ένα αντικείμενο relativedelta, για τον υπολογισμό τις χρονικής διαφοράς αρχικοποιώντας το με τις δύο παραπάνω μεταβλητές. Τέλος, στην επόμενη γραμμή, εξάγουμε τα δεδομένα του δημιουργηθέντος αντικειμένου relativedelta με τη βοήθεια της ιδιότητας __dict__ και τις τοποθετούμε συνδυαστικά σε μια συμβολοσειρά που πληροφορεί το χρήστη για τα χρόνια, τους μήνες και τις μέρες που απομένουν έως τη λήξη του domain.

```
105
106 def list_domain():
107     conn = sqlite3.connect(DB_NAME)
108     curs = conn.cursor()
109     curs.execute("SELECT domain, registrar, dns1, dns2, renew_date, cost
        FROM domains ORDER BY renew_date ASC")
110     rows = curs.fetchall()
111     if rows is None :
112         print "No domain found!"
113     else:
114         max_cols_len = []
115         for title in titles:
116             max_cols_len.append(len(title))
117         for row in rows:
118             for i in range(len(row)):
```

```
119                 if max_cols_len[i] < len(str(row[i])):
120                     max_cols_len[i] = len(str(row[i]))
121         line = ""
122         for i in range(len(titles)):
123             line = line + titles[i].ljust(max_cols_len[i]+2)
124         print line
125         print '='*len(line)
126         for row in rows:
127             line = ""
128             for i in range(len(row)):
129                 if i != 5:
130                     line = line + row[i].ljust(max_cols_len[i]+2)
131                 else:
132                     line = line + str(row[i]).rjust(max_cols_len[i])
133             print line
```

Επόμενη συνάρτηση είναι η list_domain, σκοπός της οποίας είναι η εμφάνιση μιας λίστας με όλα τα καταχωρημένα domains, μαζί φυσικά με τις πληροφορίες τους. Το ερώτημα που εκτελούμε συλλέγει όλες τις εγγραφές του πίνακα, παρατάσσοντάς τες κατά την ημερομηνία λήξης σε αύξουσα σειρά. Εδώ, αντί της fetchone καλούμε τη συνάρτηση fetchall, η οποία φέρνει όλα τα αποτελέσματα της εκτέλεσης του ερωτήματος σε έναν πολυδιάστατο πίνακα. Με τη βοήθεια τριών βρόχων, στις γραμμές 115, 117 και 118, βρίσκουμε το μέγιστο μήκος για τη κάθε στήλη του πίνακά μας, εξετάζοντας τους τίτλους καθώς κι όλα τα αποτελέσματα. Το μέγιστο μήκος το αποθηκεύουμε στον πίνακα max_cols_len και το αξιοποιούμε στη συνέχεια για να στοιχίσουμε τ' αποτελέσματα με την ljust. Τα αποτελέσματα τα ανακαλούμε εκ νέου, λίγο παρακάτω, με τη βοήθεια δύο βρόχων, η αρχή των οποίων φαίνεται στις γραμμές 126 και 128.

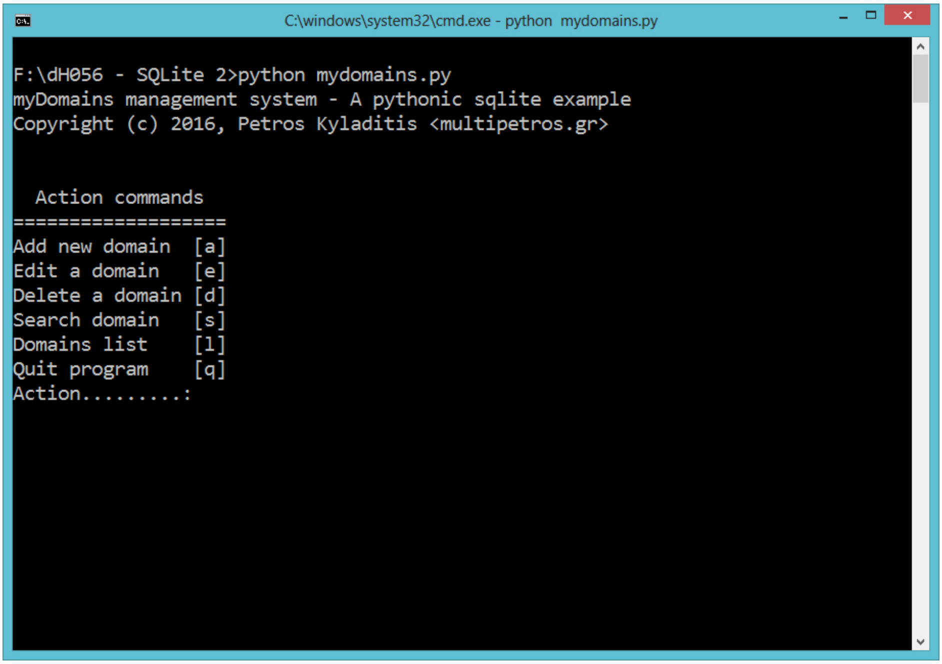
```
134
135 def mainloop():
136     action = ""
137     cmds_msg = "  Action commands\n==================\nAdd new domain
        [a]\nEdit a domain   [e]\nDelete a domain [d]\nSearch domain
        [s]\nDomains list    [l]\nQuit program    [q]"
138     while action != "q":
139         print cmds_msg
```

```
        action = raw_input("Action.........: ")
        print "\n"
        if action == "a":
            add_domain()
            raw_input("> ")
        elif action == "e":
            edit_domain()
            raw_input("> ")
        elif action == "d":
            del_domain()
            raw_input("> ")
        elif action == "s":
            search_domain()
            raw_input("> ")
        elif action == "l":
            list_domain()
            raw_input("> ")
        print "\n"
    print "Thanks for reading deltaHacker!"
    exit

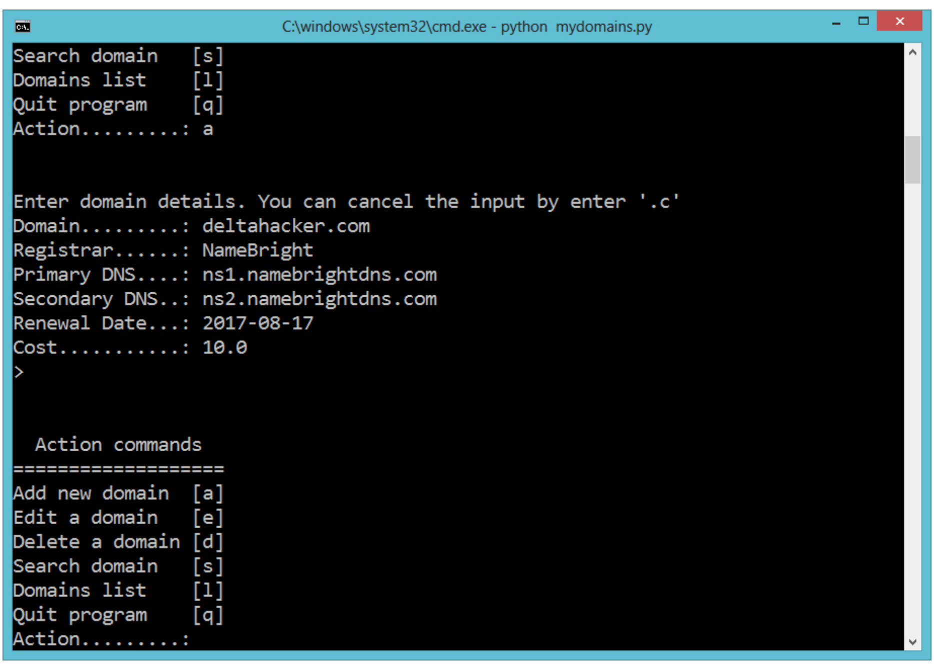
if __name__ == "__main__":
    init()
```

Η τελευταία συνάρτηση που θα εξετάσουμε αποτελεί τον κεντρικό βρόχο του προγράμματος – ή αλλιώς το βασικό του μενού. Πολύ απλά, εμφανίζει μια σειρά επιλογών και στη συνέχεια αναμένει την είσοδο του χρήστη. Ξεκινάει μια σειρά ελέγχων, όπου εξετάζεται αν η είσοδος είναι κάποια από τις προβλεπόμενες τιμές κι αν ναι τότε καλείται η σχετική μέθοδος.

Όλα όσα παρουσιάσαμε επιδέχονται βελτιώσεις και δεν αποτελούν το πιο απλό παράδειγμα. Πιστεύουμε όμως ότι η παρουσίασή μας είναι μια άριστη ευκαιρία ώστε να εξοικειωθούμε με τη χρήση βάσεων δεδομένων σε εφαρμογές μας, χωρίς την ανάγκη για παρουσία server. Όπως πάντα σας προτρέπουμε να πειραματιστείτε άφοβα, δοκιμάζοντας όχι μόνο να τροποποιήσετε αλλά και να μεταφέρετε το πρότζεκτ με τα domains στις αγαπημένες σας γλώσσες προγραμματισμού, όποιες κι αν είναι αυτές. Σε κάθε περίπτωση, ολόκληρο τον κώδικα του παρόντος άρθρου θα τον βρείτε στο http://bit.ly/dH056mydomains. Καλή σας διασκέδαση!




*Κατά την έναρξη του προγράμματος εμφανίζεται ένα μήνυμα καλωσορίσματος και φυσικά το κεντρικό μενού επιλογών. Ίσως σας φαίνεται παλιομοδίτικη προσέγγιση, εμάς πάντως μας αρέσει πολύ αυτός ο ρετρό χαρακτήρας.*




*Προσθέτοντας ένα νέο –κι εντελώς τυχαίο– domain.*

C:\windows\system32\cmd.exe - python mydomains.py

```
>


  Action commands
===================
Add new domain   [a]
Edit a domain    [e]
Delete a domain  [d]
Search domain    [s]
Domains list     [l]
Quit program     [q]
Action.........: s


Domain to show: parabing.com


Domain.........: parabing.com
Registrar......: gandi
Primary DNS....: NS1.BOX.COLDER.XYZ
Secondary DNS..: NS6.GANDI.NET
Renewal Date...: 2017-01-17 [0 years, 5 months, 0 days left]
Cost...........: 9.9
>
```

*Η λειτουργία αναζήτησης επιστρέφει άμεσα τα αποτελέσματα στην οθόνη μας, καθώς και το χρονικό διάστημα έως την επόμενη ανανέωση του domain (σε χρήσιμη μορφή).*

C:\windows\system32\cmd.exe - python mydomains.py

```


  Action commands
===================
Add new domain   [a]
Edit a domain    [e]
Delete a domain  [d]
Search domain    [s]
Domains list     [l]
Quit program     [q]
Action.........: e


Domain to edit: deltahacker.gr
Domain 'deltahacker.gr' not found!
Domain to edit: deltahacker.com
Enter new data, or leave it blank to keep the existed values
Domain        [deltahacker.com]:
Registrar     [NameBright]: gandi
Primary DNS   [ns1.namebrightdns.com]: NS1.GANDI.NET
Secondary DNS [ns2.namebrightdns.com]: NS6.GANDI.NET
Renewal Date  [2017-08-17]:
Cost          [10.0]:
>
```

*Μετά την αλλαγή καταχωρητή, επεξεργαστήκαμε και τις ήδη καταχωρημένες πληροφορίες για το domain μας.*

**Skill:** Beginner
**Tags:** BIOS, UEFI, boot process, bootloader, GRUB, init, sysv, upstart, systemd

# Πώς ξεκινούν τα PC και πώς φορτώνει το Linux

**Έχετε αναρωτηθεί τι συμβαίνει όταν πατάτε το κουμπί power-on του υπολογιστή; Σας έχει ποτέ απασχολήσει η διαδικασία boot του Linux; Τι είναι αυτό το systemd για το οποίο έτυχε να διαβάσετε κάπου στο Web; Όσο απλοϊκά ή αδιάφορα κι αν φαίνονται ερωτήματα σαν τα προηγούμενα, υποψιαζόμαστε ότι κάποιοι δεν μπορείτε να δώσετε ικανοποιητικές απαντήσεις. Αλλά για μια στιγμή. Χρειάζεται να σκοτίζεστε με τέτοια θέματα;**

*του Χ.Β.*

Γενικά, όχι, δεν χρειάζεται. Αν όμως θέλετε να λέτε ότι είστε καλός system administrator και πράγματι να είστε καλός system administrator, τότε θεωρούμε πως, ναι, οφείλετε να σκοτίζεστε με τέτοια θέματα – έστω κι αν είναι για μια φορά στο τόσο. Πιθανώς εξάλλου ν' αντιμετωπίσετε παρόμοια ερωτήματα στο πλαίσιο εξετάσεων για κάποια πιστοποίηση που προσπαθείτε να πάρετε. Κρίμα δεν θα ήταν να δίνατε ελλιπή απάντηση, μόνο και μόνο επειδή το ερώτημα πάντα σας φαινόταν αδιάφορο ή προφανές; Αλλά ας μην αγχωνόμαστε τώρα με εξετάσεις και πιστοποιήσεις. Αντίθετα, ελάτε να συζητήσουμε για το τι συμβαίνει σ' έναν υπολογιστή με το που τον ανοίγουμε, μετά να μάθουμε και πώς ακριβώς φορτώνει το Linux. Κάπως έτσι θα φτάσουμε στη διεργασία init και θα γνωρίσουμε το systemd, το οποίο όλες οι σύγχρονες διανομές έχουν υιοθετήσει.

## Το boot process από κοντά

Ακόμη κι όσοι γνωρίζουν καλά τι συμβαίνει από το power-on ως το περιβάλλον γραφικών ή έστω ως την κονσόλα απλού κειμένου με την προτροπή για login, σπάνια σκέφτονται για τις σχετικές παρασκηνιακές διαδικασίες. Βέβαια εκείνοι που ξέρουν και γνωρίζουν δικαιούνται να μη σκέφτονται *κάθε φορά* για το τι συμβαίνει. Κέρδισαν αυτό το δικαίωμα από τη στιγμή που αφιέρωσαν χρόνο για να διαβάσουν,

να μελετήσουν και να πειραματιστούν με τα PC τους, οπότε ας τους αφήσουμε ήσυχους, εκεί, στον άνετο καναπέ τους ή στον επιβλητικό τους θρόνο ή στην όποια καταλληλότερη μεταφορά μπορείτε να σκεφτείτε, τέλος πάντων. Η αλήθεια είναι ότι μερικές φορές έχουν ένα κάπως μπλαζέ υφάκι, κάπως ενοχλητικό ομολογουμένως. Λοιπόν! Προτείνουμε να μάθουμε κι εμείς για το boot process, αν μη το άλλο για να δούμε τι προκαλεί αυτό το υφάκι.

Η διαδικασία εκκίνησης ενός οποιουδήποτε OS ξεκινά με λειτουργίες που επιτελούνται αρχικά από το BIOS ή το UEFI του υπολογιστή και μετά από τον bootloader. Στη συνέχεια φορτώνει ο πυρήνας (kernel) και, ειδικά στην περίπτωση του Linux, η όλη διαδικασία ολοκληρώνεται με τη βοήθεια της διεργασίας init. Σε γενικές γραμμές, λοιπόν, μπορούμε να μιλάμε για τέσσερα διαφορετικά στάδια: BIOS/UEFI, bootloader, kernel και init.

**Στάδιο 1α – BIOS.** Το Basic Input/Output System είναι το κλασικό firmware στην πλατφόρμα του PC. Αν κάποια στιγμή χρειάστηκε ν' αλλάξετε τη σειρά αναζήτησης bootable code στους δίσκους (εσωτερικούς ή εξωτερικούς) του PC σας, τότε έχετε ήδη αλληλεπιδράσει με το BIOS. Πρώτη δουλειά του BIOS με το που ανοίγουμε τον υπολογιστή είναι να εκτελεί μια σειρά από βασικούς διαγνωστικούς ελέγχους, ώστε να διαπιστωθεί αν το hardware βρίσκεται σε μια στοιχειωδώς καλή κατάσταση. Αφού το BIOS δει τους δίσκους, ψάχνει για το λεγόμενο boot block. Το πού ψάχνει σε κάθε δίσκο εξαρτάται από τη λογική διευθέτησης των κατατμήσεών του (MBR ή GPT partitioning). Πιο συγκεκριμένα, αν το partition table του υπό εξέταση δίσκου ακολουθεί τη λογική του MBR, τότε ψάχνει για το Master Boot Record. Αυτό βρίσκεται στον πρώτο τομέα (sector) του δίσκου, συγκεκριμένα στα πρώτα 512 bytes. Το MBR ενδέχεται να περιλαμβάνει κώδικα για το φόρτωμα κάποιου άλλου προγράμματος, κάπου αλλού στο δίσκο. Το τελευταίο αυτό πρόγραμμα είναι εκείνο που τελικά φορτώνει τον bootloader του λειτουργικού συστήματος.

**Στάδιο 1β – UEFI.** Βελτιωμένος απόγονος του παλιού καλού BIOS είναι το UEFI (Unified Extensible Firmware Interface). Η πλειονότητα των σύγχρονων PCs έχει firmware με UEFI. Η Apple το υιοθέτησε για τους υπολογιστές της χρόνια νωρίτερα, ωστόσο και τα Windows πλέον υποστηρίζουν την εκκίνηση από UEFI. Αν και το GPT partitioning σχεδιάστηκε ώστε να χρησιμοποιείται σε υπολογιστές με UEFI, αξίζει να σημειώσουμε ότι το booting από δίσκους GPT είναι δυνατό ακόμη και σε PCs με το κλασικό BIOS: ο πρώτος τομέας ενός δίσκου GPT είναι δεσμευμένος για κώδικα MBR, ώστε η εκκίνηση να 'ναι δυνατή και σε υπολογιστή χωρίς UEFI.

**Στάδιο 2 – Bootloader.** Η αποστολή του bootloader είναι να εκκινεί το λειτουργικό σύστημα, ασχέτως αν πρόκειται για το Linux ή για κάποιο άλλο. Ειδικά στην περίπτωση του Linux, ο bootloader επιτρέπει την επιλογή πυρήνα προς φόρτωμα, ενώ είναι και ικανός να του περνά μια σειρά από παραμέτρους. Ο bootloader που χρησιμοποιείται σήμερα κατά κόρον στον κόσμο του Linux είναι ο GRUB.

**Στάδιο 3 – Πυρήνας (kernel).** Αφού ο bootloader δώσει στον πυρήνα τις όποιες παραμέτρους για το boot, πρέπει να τον φορτώσει στη μνήμη. Αυτό είναι λίγο δυσκολότερο απ' όσο ακούγεται. Βλέπετε, αν κι ο πυρήνας διαχειρίζεται το hardware του υπολογιστή, τη στιγμή του boot δεν είναι όλοι οι οδηγοί συσκευών διαθέσιμοι και συνεπώς κάποια υποσυστήματα είναι σαν να μην υπάρχουν. Το πρόβλημα αντιμετωπίζεται προσαρτώντας ένα προσωρινό root filesystem, το οποίο περιλαμβάνει τα αρθρώματα (modules) που χρειάζεται ο πυρήνας προκειμένου να δει το hardware.

Σε παλιότερες εκδόσεις του Linux, το προσωρινό αυτό filesystem αποτελούσε το αρχείο ονόματι initrd (INITial Ram Disk). Με την προσάρτηση του initrd ο πυρήνας αποκτούσε όλους τους απαραίτητους οδηγούς για την επιτυχή ολοκλήρωση της εκκίνησης. Μόλις φόρτωνε ό,τι χρειαζόταν, το initrd αποπροσαρτόταν και γινόταν η προσάρτηση του root filesystem, στον δίσκο της εγκατάστασης. Στις σύγχρονες διανομές Linux, τη θέση του αρχείου initrd έχει πάρει το initramfs. Κι αυτό είναι ένα προσωρινό σύστημα αρχείων, μόνο που τώρα δεν αποτελεί ξεχωριστό αρχείο αλλά είναι ενσωματωμένο στον πυρήνα. Φυσικά, περιλαμβάνει όλους τους απαραίτητους οδηγούς ώστε να είναι δυνατή η προσάρτηση του root filesystem.

Χάρη στο initrd ή στο initramfs, ο πυρήνας αποκτά πρόσβαση στα αρθρώματα που χρειάζεται προκειμένου να δημιουργήσει ένα root device για την προσάρτηση της ριζικής κατάτμησης (root partition). Αρχικά, η προσάρτησή της γίνεται με δικαιώματα ανάγνωσης μόνο (read only mode). Έτσι, το εργαλείο ελέγχου ακεραιότητας κατατμήσεων ονόματι fsck τρέχει με ασφάλεια. Αφού ολοκληρώσει τους ελέγχους του, η ριζική κατάτμηση αποπροσαρτάται και προσαρτάται εκ νέου – αλλά αυτή τη φορά τόσο με δικαιώματα ανάγνωσης όσο και εγγραφής (read-write mode). Χωρίς να χάσει χρόνο, ο πυρήνας αναζητά το πρόγραμμα init και το ενεργοποιεί.

**Στάδιο 4 – Init.** Το init στο Linux είναι πάντοτε η πρώτη διεργασία που ενεργοποιείται. Αποστολή της είναι η εκκίνηση όλων των απαραίτητων υπηρεσιών του λειτουργικού. Τώρα, το πώς τις ξεκινά εξαρτάται από την υλοποίηση. Ναι, δεν υπάρχει ένα μόνο init στο Linux. Για την ακρίβεια, διατίθενται τρεις βασικές υλοποιήσεις.

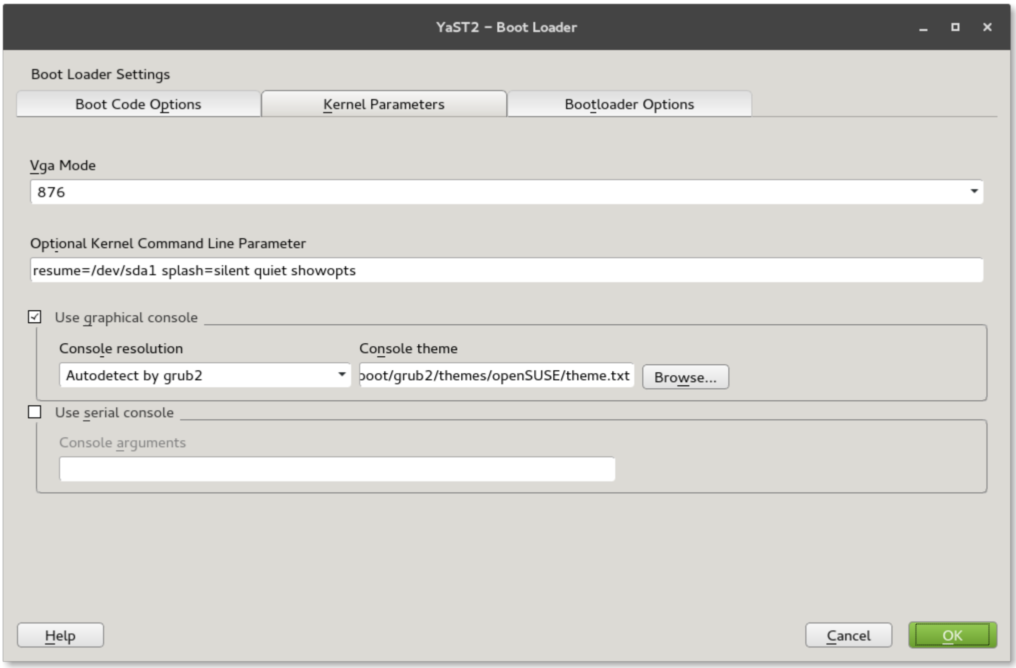
**System V init (sysv).** Πρόκειται για το παραδοσιακό σύστημα init στον κόσμο του Linux, το οποίο βασιζόμενο στα λεγόμενα startup scripts ξεκινά/σταματά άλλες διεργασίες με τη σειρά. Η κατάσταση του συστήματος καθορίζεται από τα runlevels, με κάθε runlevel να ξεκινά/σταματά το σύστημα κατά τρόπο διαφορετικό.

**Upstart.** Είναι το σύστημα init που χρησιμοποιούταν σε παλαιότερες εκδόσεις του Ubuntu Linux. Στο upstart έχουμε εργασίες (jobs) και γεγονότα (events), όπου κάθε εργασία ξεκινά κι εκτελεί συγκεκριμένες ενέργειες ανταποκρινόμενη σε γεγονότα.

**Systemd.** Είναι το νέο στάνταρτ για το init στο Linux. Κεντρική ιδέα στο systemd είναι οι στόχοι (goals). Ουσιαστικά, υπάρχουν στόχοι που το systemd προσπαθεί να πετύχει ικανοποιώντας τις εξαρτήσεις (dependencies) κάθε στόχου.

## Η συνέχεια

Ως εδώ έχουμε αποκτήσει μια πολύ καλή εικόνα για το πώς ξεκινά ο υπολογιστής μας. Με δεδομένο εξάλλου ότι η πλειονότητα των διανομών Linux έχουν υιοθετήσει το Systemd, αξίζει να εστιάσουμε σ' αυτό. Όπως θα διαπιστώσετε κι εσείς, το Systemd κάνει πολλά περισσότερα από το να ξεκινά υπηρεσίες του λειτουργικού. Είστε διαχειριστής συστήματος που σέβεται τον εαυτό του, οπότε οφείλετε να μάθετε περισσότερα γι' αυτό. Δεν θα το μετανιώσετε. Γυρίστε τώρα στη σελίδα 64.




*Οι παράμετροι που περνά ο bootloader (GRUB) στον πυρήνα ενός συστήματος με openSUSE, όπως φαίνονται μέσα από το εργαλείο διαχείρισης YaST.*

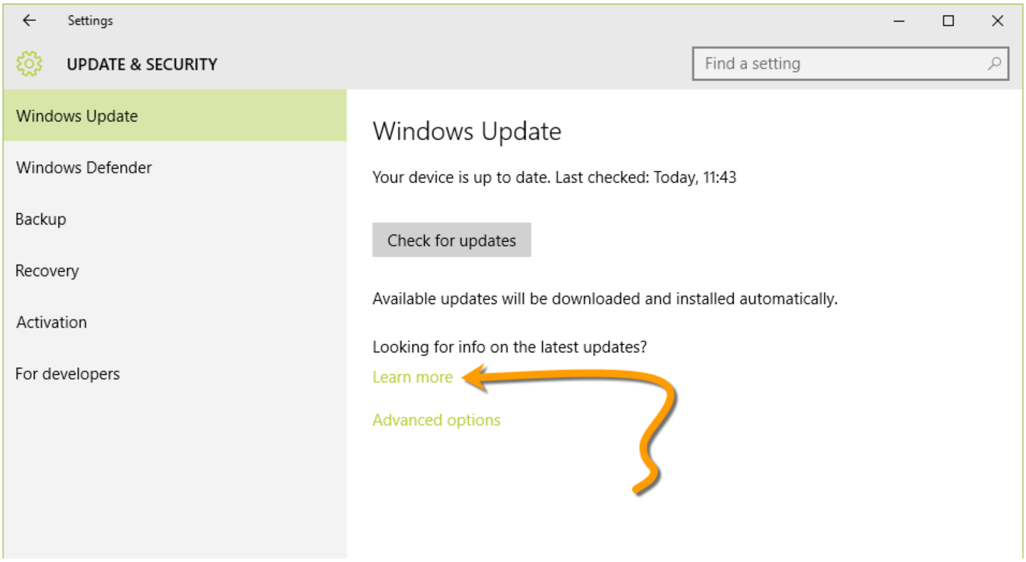
# Επετειακά Windows 10 μαζί με BASH

**Κυκλοφόρησε στις αρχές Αυγούστου το λεγόμενο Anniversary Update για τα Windows 10, την έκδοση των Windows που σχεδόν όλοι αγαπούν αλλά πολύ λιγότεροι το παραδέχονται δημόσια. Σας δείχνουμε πώς ν' αναβαθμιστείτε στο Anniversary Update χωρίς να περιμένετε, αλλά και πώς να χαρίσετε στα Windows 10 το BASH. Ναι, το BASH του Linux. Τώρα μπορείτε να το έχετε *και* στα Windows.**

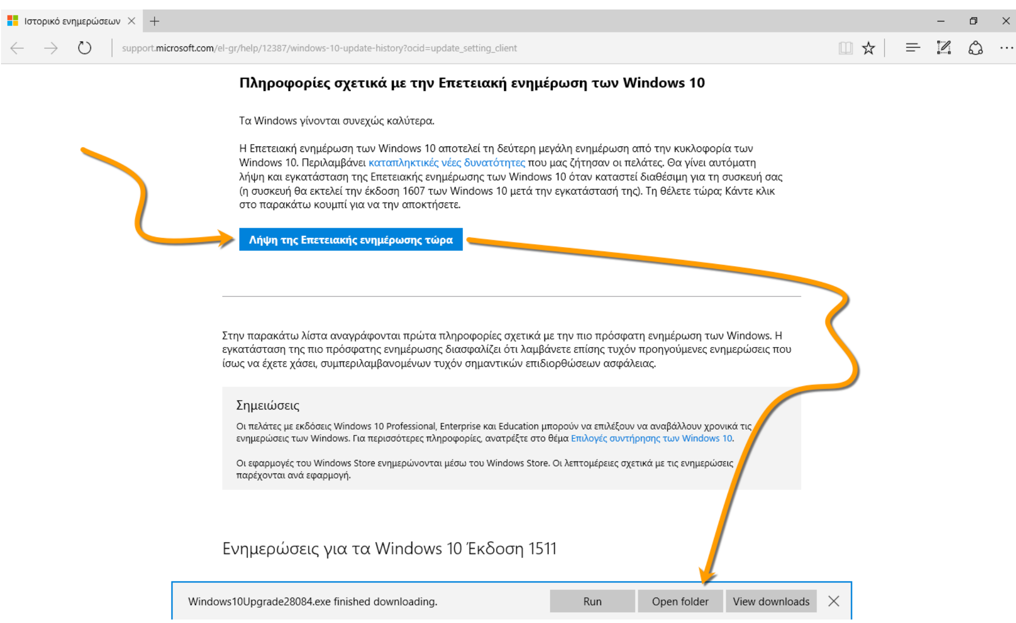
Το Anniversary Update φέρνει νέα χαρακτηριστικά για την Cortana, το Windows Ink και τον Windows Defender. Ο δε Edge browser καταναλώνει τώρα λιγότερη ενέργεια, ενώ υποστηρίζει και extensions. Μετά την αναβάθμιση οι χρήστες του λειτουργικού έχουν πρόσβαση στο Skype Preview, ενώ και οι gamers απολαμβάνουν νέες δυνατότητες όπως, π.χ., το εμπλουτισμένο σετ φωνητικών εντολών για το Xbox. Μάθετε περισσότερα για το Anniversary Update παρακολουθώντας το σχετικό video στο https://youtu.be/80Qz0Gbjh6Q.

## Αναβάθμιση χωρίς αναμονή

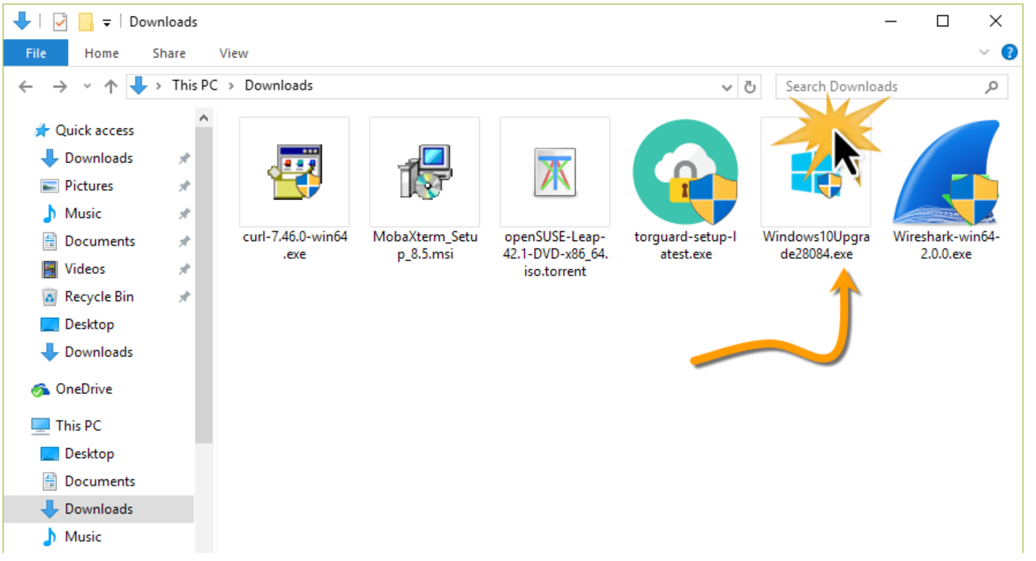
Για τη λήψη του Anniversary Update δεν χρειάζεται κάτι από πλευράς χρήστη. Απλά, επειδή η διάθεση της αναβάθμισης δεν γίνεται ταυτόχρονα σ' όλον τον πλανήτη αλλά σταδιακά, ίσως χρειαστεί να περιμένετε λίγο. Υπάρχει πάντως δυνατότητα να πάρετε άμεσα το Anniversary Update, με λίγα μόλις κλικ. Δείτε τα screenshots που ακολουθούν, διαβάστε και τις αντίστοιχες περιγραφές.



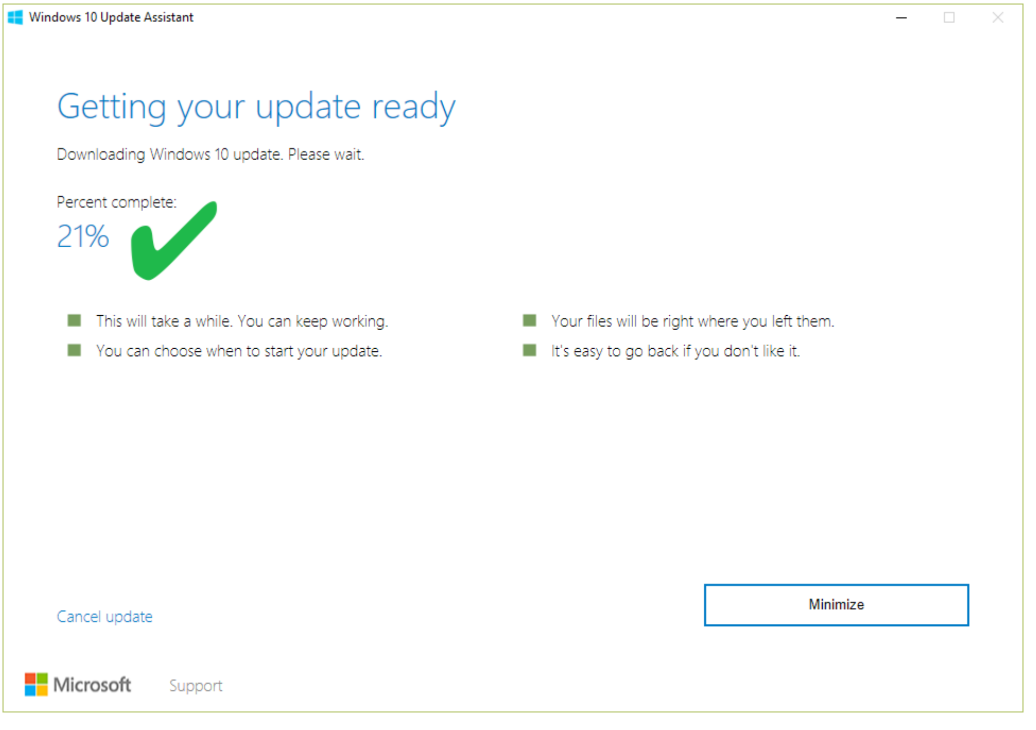

*Ξεκινάμε πατώντας στο Windows key. Επιλέγουμε Settings και μετά Update & Security, ακολούθως κάνουμε κι ένα κλικ στο Learn more.*




*Ανοίγει τότε ένα παράθυρο του Edge browser. Πατάμε στο μεγάλο μπλε κουμπί που γράφει "Λήψη της επετειακής ενημέρωσης τώρα". Θα ξεκινήσει το κατέβασμα ενός εκτελέσιμου με το Anniversary Update. Όταν το downloading ολοκληρωθεί, κάνουμε ένα κλικ στο κουμπί Open folder (βλ. κάτω μέρος του Edge).*




*Αποθηκευμένο στον φάκελο Downloads του PC μας είναι το εκτελέσιμο ονόματι Windows10upgrade28084.exe. Πρόκειται για το Anniversary Update και το εκτελούμε με κλικ πάνω στο εικονίδιό του.*



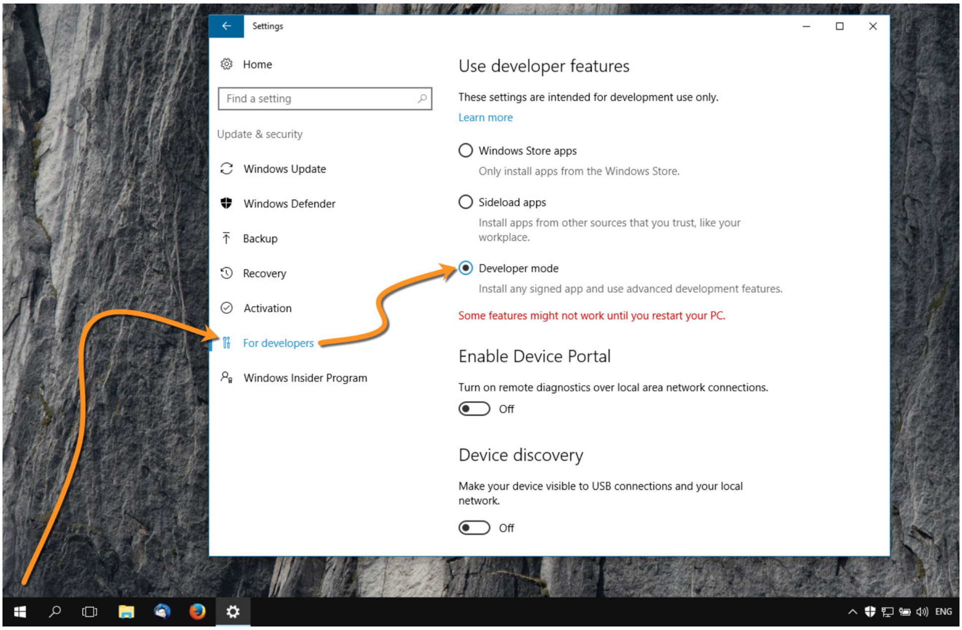

*Για την ολοκλήρωση της εφαρμογής του Anniversary Update θα χρειαστεί λίγη υπομονή και κάποια στιγμή θα γίνει επανεκκίνηση του μηχανήματος. Πέρα απ' αυτό, η όλη διαδικασία θα προχωρήσει ομαλά – ή τουλάχιστον αυτό συνέβη στην περίπτωσή μας.*

## Τι είναι το BASH για τα Windows

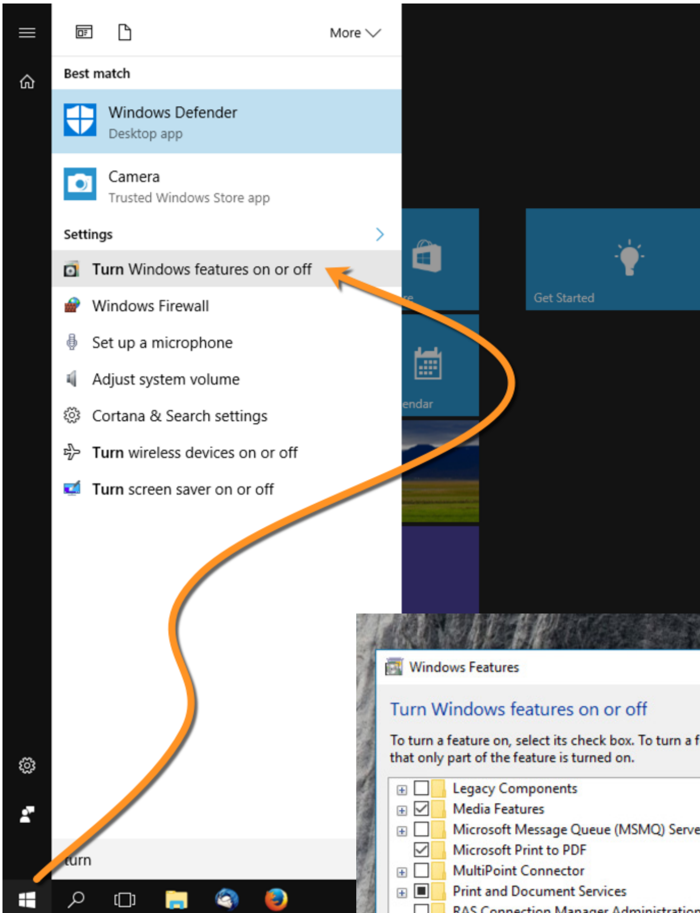
Μεταξύ των νέων χαρακτηριστικών που φέρνει το Anniversary Update είναι το λεγόμενο Windows Subsystem for Linux. Ουσιαστικά η Microsoft έχει συνεργαστεί με την Canonical, τη μητέρα του Ubuntu, ώστε μετά το Anniversary Update να παρέχεται επιλογή εγκατάστασης του BASH. Η συγκεκριμένη εκδοχή του δημοφιλούς κελύφους τρέχει πάνω από ένα συμβατό με τον πυρήνα του Linux ενδιάμεσο (interface), το οποίο αξίζει να σημειωθεί ότι δεν περιλαμβάνει καθόλου κώδικα από το Linux: είναι απλά συμβατό με τον πυρήνα σε επίπεδο API. Διευκρινίζοντας λίγο περισσότερο, να πούμε ότι το BASH για τα Windows δεν αποτελεί περιβάλλον chroot, ούτε είναι container, ούτε εικονική μηχανή. Σίγουρα δεν είναι, τέλος, κάποιου είδους lightweight-ή-όχι διανομή. Επιτρέπει, ωστόσο, την εκτέλεση εντολών του BASH αλλά και προγραμμάτων Linux για την κονσόλα απλού κειμένου, παρέχοντάς τους πρόσβαση στο filesystem των Windows.

## Windows 10 με ολίγη από Linux

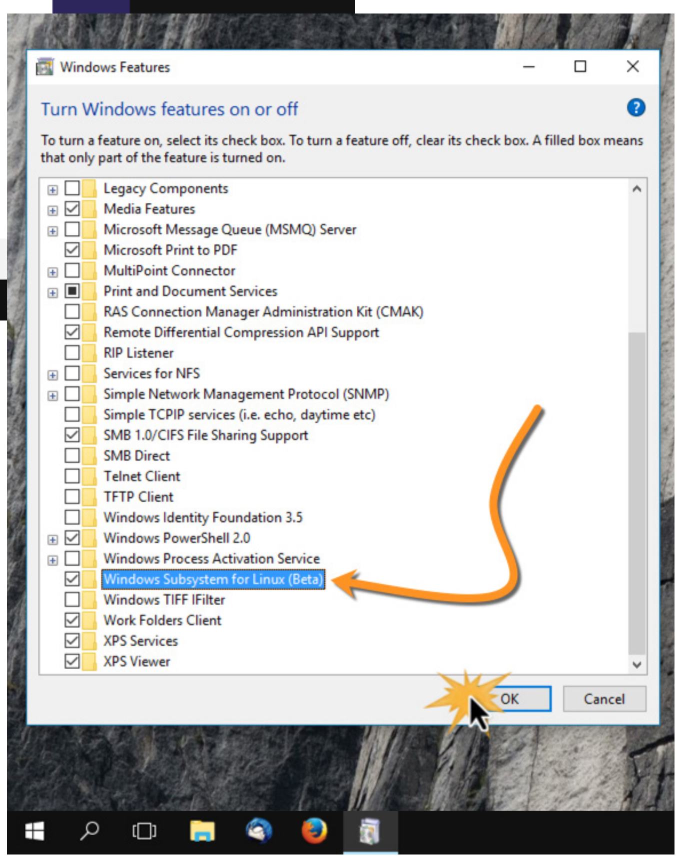
Για την ενεργοποίηση του BASH στα Windows 10 χρειάζεται να έχουμε PC αρχιτεκτονικής 64bit, την 64μπιτη εκδοχή των Windows 10 εγκατεστημένη, καθώς και το Anniversary Update εφαρμοσμένο. Δείτε τα screenshot που ακολουθούν και διαβάστε τις αντίστοιχες περιγραφές.



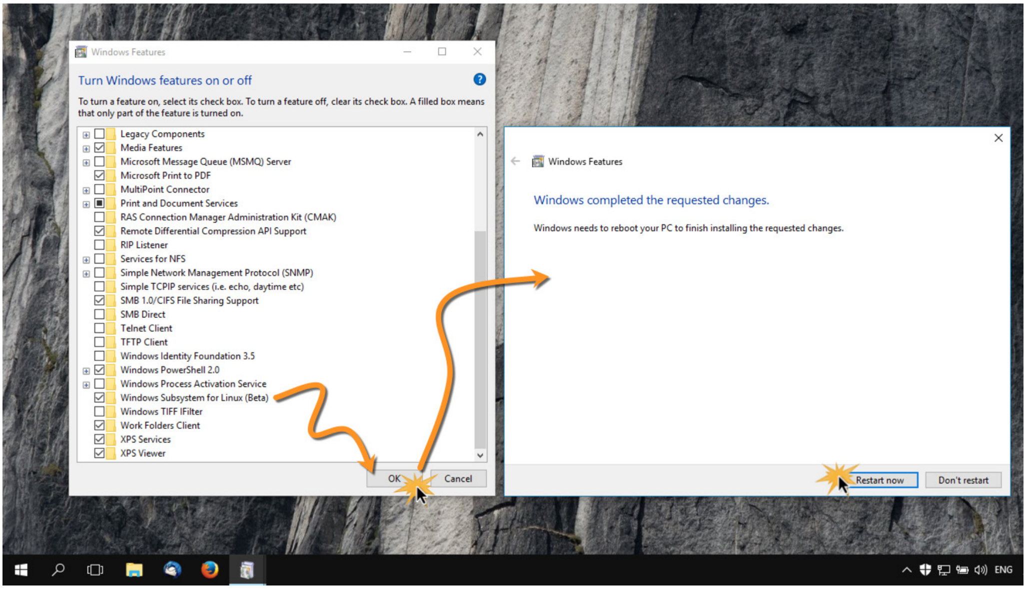

*Ξεκινάμε πατώντας στο Windows key, μετά στο Settings. Πηγαίνουμε στο Update & Security, κάνουμε ένα κλικ στην επιλογή For developers (αριστερά) κι άλλο ένα στο radio button με όνομα Developer mode (δεξιά). Αμέσως εμφανίζεται ένα pop-up προειδοποιώντας μας ότι η ενεργοποίηση του Developer mode είναι επικίνδυνη ενέργεια που ίσως προκαλέσει θερμοπυρηνική σύρραξη, κάπου στον πλανήτη. Απρόθυμα δεχόμαστε το ρίσκο με κλικ στο κουμπί Yes. Μετά λίγα δευτερόλεπτα ειδοποιούμαστε ότι για την ενεργοποίηση συγκεκριμένων χαρακτηριστικών χρειάζεται ένα reboot. Ανακουφισμένοι που δεν έχουμε ακόμη επεισόδια, το κάνουμε.*



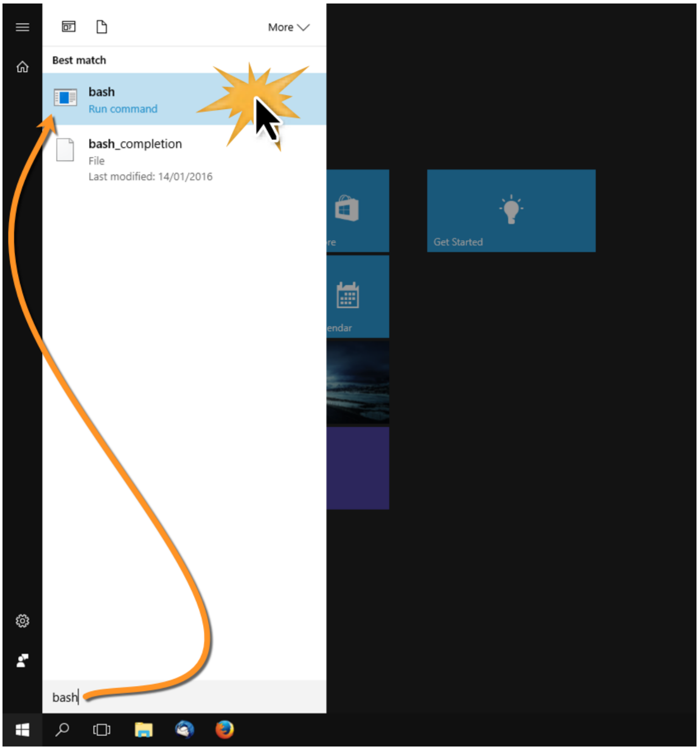

*Μετά την επανεκκίνηση πατάμε (ξανά) στο Windows key κι αρχίζουμε να πληκτρολογούμε "turn windows" (χωρίς) τα εισαγωγικά. Θα παρουσιαστεί η επιλογή Turn Windows features on or off και θα κάνουμε ένα κλικ πάνω της.*



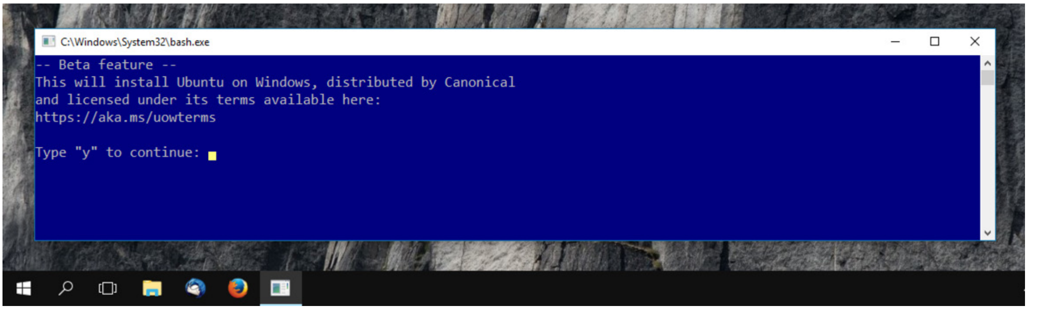

*Εμφανίζεται το παράθυρο με τίτλο Windows Features. Σκρολάρουμε τα περιεχόμενά του προς τα κάνω και τσεκάρουμε την επιλογή Windows Subsystem for Linux (Beta). Επικυρώνουμε με κλικ στο κουμπί OK, κάτω.*



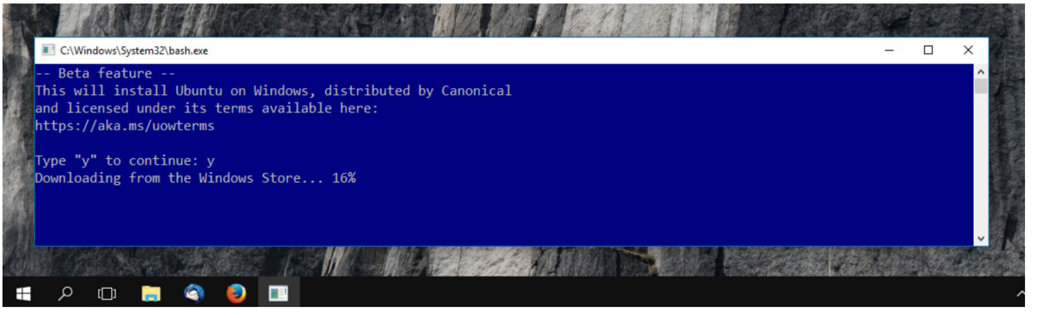

*Και μετά το προηγούμενο τσεκάρισμα, από ένα νέο παράθυρο τα Windows μάς ενημερώνουν ότι θέλουν reboot. Ναι, πάλι. Στα Windows αρέσουν τα reboot. Πολύ.*



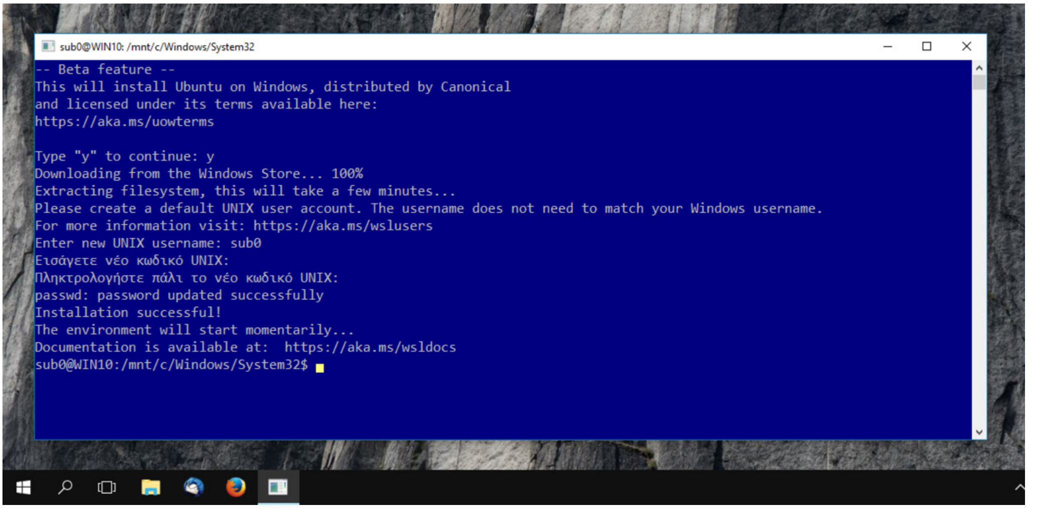

*Γίνεται και η δεύτερη επανεκκίνηση, πατάμε στο Windows key, γράφουμε "bash" (χωρίς τα εισαγωγικά), στη γραμμή bash / Run command, πάνω πάνω, κάνουμε ένα κλικ.*




*Ανοίγει τότε ένα παράθυρο Command Prompt των Windows και πληροφορούμαστε ότι πρόκειται να εγκατασταθεί το Ubuntu. Υπενθυμίζουμε ότι δεν πρόκειται να δημιουργηθεί κάποιο chroot περιβάλλον, ούτε κάποιου είδους εικονική μηχανή. Απλά η διατύπωση των ανθρώπων της Microsoft δεν είναι και η ακριβέστερη δυνατή, αλλά δεν είναι και η πρώτη φορά που ακούμε ή διαβάζουμε ανακριβή πράγματα. Για τη συνέχεια πατάμε το πλήκτρο [Y] – βεβαίως κι ένα [Enter].*



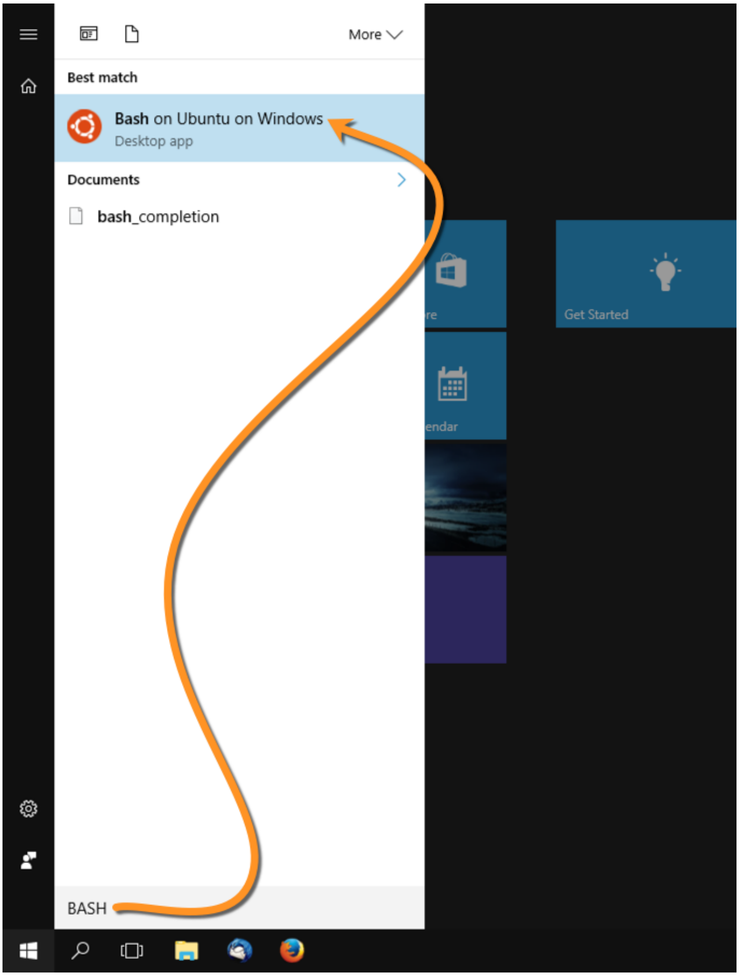

*Ακολουθεί το κατέβασμα, από το Windows Store παρακαλούμε, όλων των απαραίτητων βιβλιοθηκών κι εργαλείων για τη φιλοξενία του BASH στα Windows.*



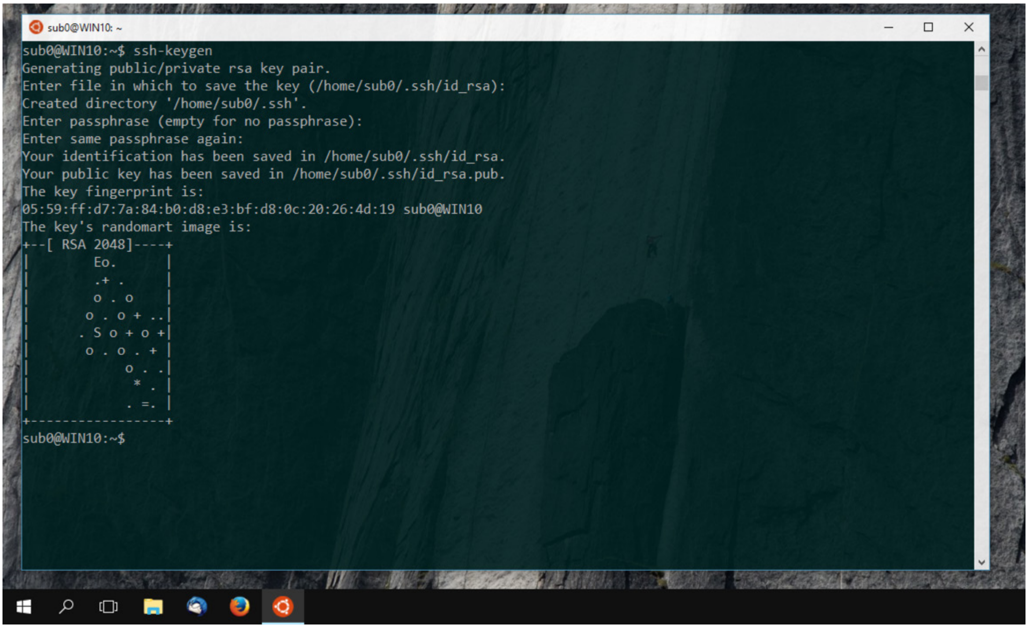

*Μετά την ολοκλήρωση της εγκατάστασης του BASH καλούμαστε να δημιουργήσουμε το λογαριασμό ενός νέου χρήστη. Το σχετικό username δεν χρειάζεται να 'ναι ίδιο μ' εκείνο του χρήστη μας στα Windows. Μετά την πληκτρολόγηση username δίνουμε κι ένα password – δύο φορές, για λόγους ασφαλείας. Ύστερα κι απ' αυτό είμαστε έτοιμοι για το BASH στα Windows μας!*

## Τι κάνουμε με το BASH για Windows;

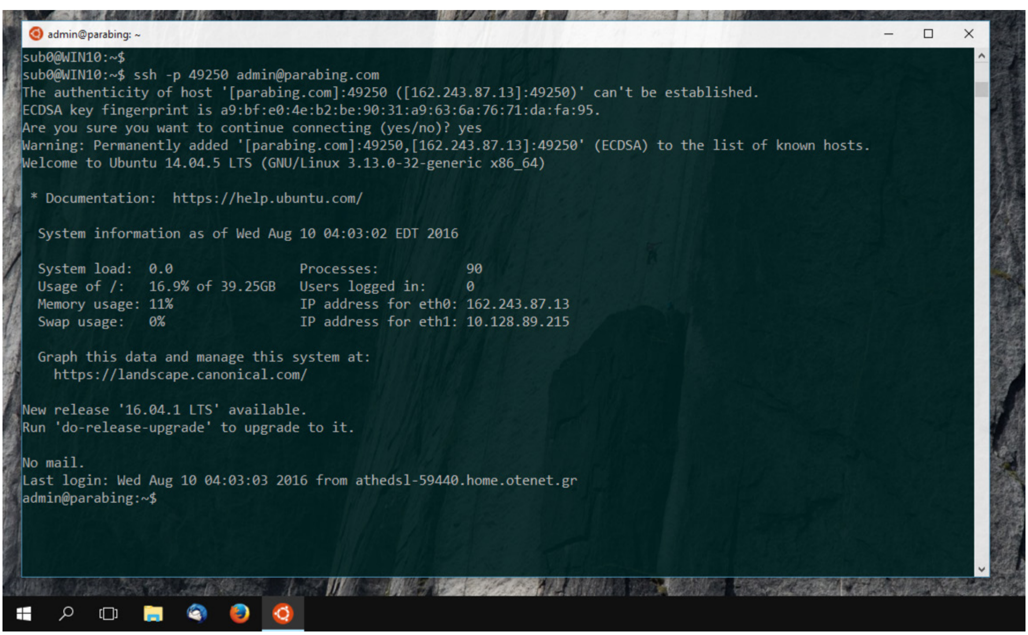
Έχουμε δυνατότητα εκτέλεσης όλων των εντολών που γνωρίζει το BASH, καθώς κι οποιασδήποτε text-based εφαρμογής που διατίθεται για το Ubuntu. Οι επεξεργαστές απλού κειμένου nano, vim και emacs είναι τα πρώτα παραδείγματα που μας έρχονται κατά νου. Αν κάποια εφαρμογή δεν είναι εγκατεστημένη, όπως, π.χ., συμβαίνει με το htop, την εγκαθιστούμε όπως ακριβώς θα την εγκαθιστούσαμε από ένα τερματικό του Ubuntu. Και ναι, εννοείται ότι μπορούμε να δίνουμε εντολές όπως sudo apt-get update, sudo apt-get upgrade, sudo apt-get install κ.ο.κ. Ένα πολύ χρήσιμο εργαλείο που έχουμε στη διάθεσή μας είναι το ssh (δηλαδή ο επίσημος SSH client του πακέτου OpenSSH). Μόνο και μόνο γι' αυτό, νομίζουμε ότι αξίζει να εγκαταστήσει κανείς το BASH στα Windows 10. Δείτε τα τρία ακόλουθα screenshots, διαβάστε τις αντίστοιχες περιγραφές – και καλορίζικα τα Linux-friendly Windows σας!



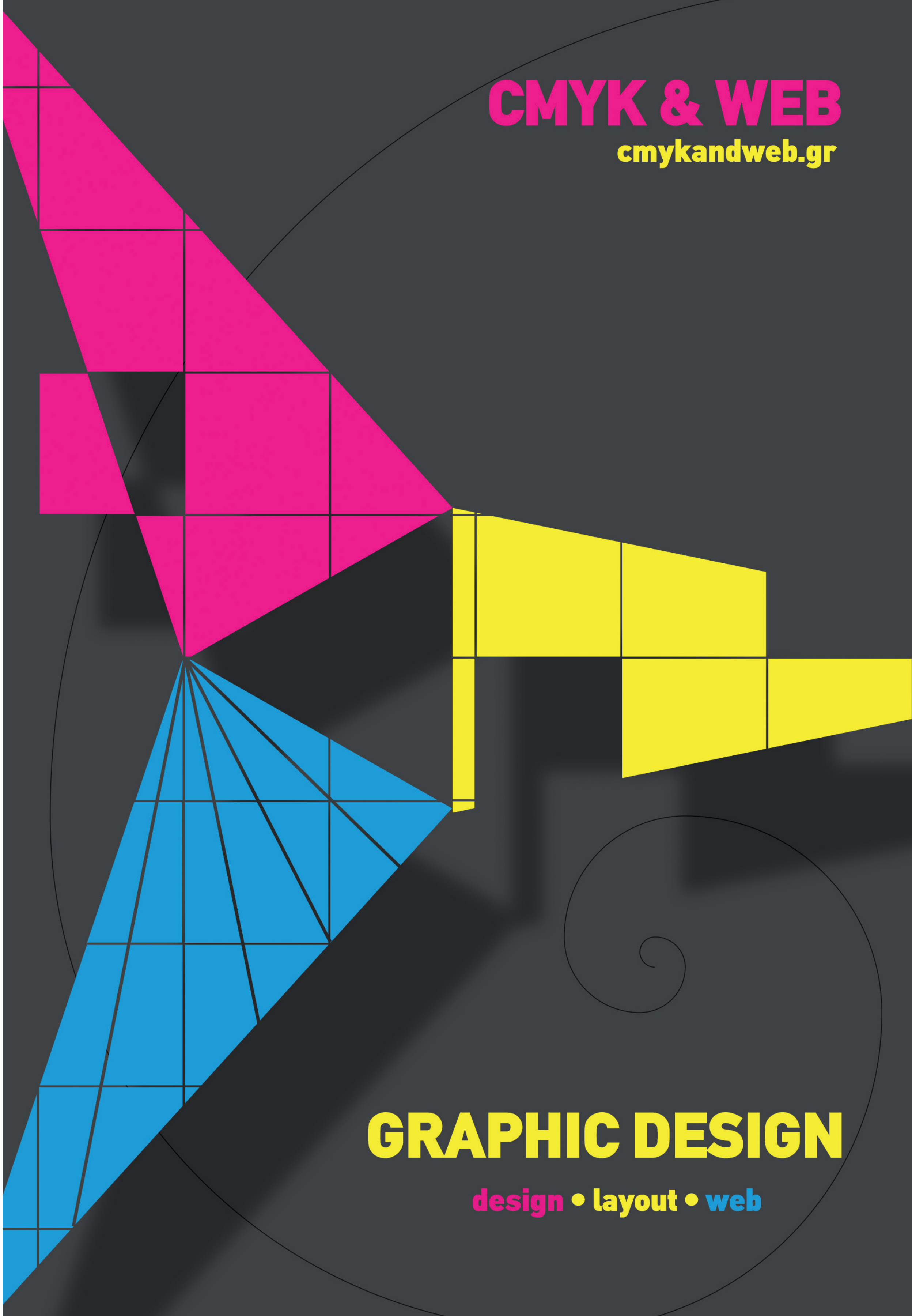

*Κάθε φορά που θέλουμε ένα τερματικό με το κέλυφος του BASH, ένας τρόπος να το έχουμε είναι με κλικ στο Windows key, πληκτρολόγηση του "bash" (χωρίς τα εισαγωγικά) και κλικ πάνω στο αποτέλεσμα Bash on Ubuntu on Windows. Μπορούμε, φυσικά, να καρφιτσώσουμε το αντίστοιχο εικονίδιο στο μενού Start ή στην taskbar.*




*Όπως θα κάναμε σε μια διανομή Linux, έτσι κι εδώ φτιάχνουμε ένα ζεύγος ιδιωτικού-δημοσίου κλειδιού για το SSH και για το λογαριασμό χρήστη που δημιουργήσαμε αμέσως μετά την εγκατάσταση του BASH. Το δημόσιο κλειδί μπορούμε να το στέλνουμε σε απομακρυσμένους hosts στους οποίους έχουμε πρόσβαση, ώστε να συνδεόμαστε σ' αυτούς χωρίς πληκτρολόγηση password. Διαβάστε περισσότερα στο https://deltahacker.gr/?p=13357.*




*Σύνδεση σε απομακρυσμένο SSH server από τον SSH client που έχει το BASH του Ubuntu, κάτω από τα Windows 10. Έτσι όπως το βλέπουμε εμείς, μόνο και μόνο γι' αυτή τη δυνατότητα αξίζει να εγκαταστήσουμε το BASH. (Αλήθεια, θυμάται κανείς πότε ενημερώθηκε τελευταία φορά το PuTTY για Windows;)*


CMYK & WEB
cmykandweb.gr
GRAPHIC DESIGN
design • layout • web

**Skill:** Beginner
**Tags:** SSH, filesystem, user space

# Απομακρυσμένη προσάρτηση μέσω SSH

**Το Filesystem in Userspace ή απλά FUSE είναι ένα ενδιάμεσο λογισμικού που επιτρέπει σε χρήστες *χωρίς* δικαιώματα διαχειριστή να ρυθμίζουν και να προσαρτούν συστήματα αρχείων. Στο παρόν άρθρο δείχνουμε πώς με τη βοήθεια των FUSE και SSHFS προσαρτούμε καταλόγους σε απομακρυσμένους servers, μέσω του ασφαλούς πρωτοκόλλου SSH.**

Το FUSE διατίθεται για Linux, *BSD, OS X και άλλα Unix-οειδή λειτουργικά. Πιθανότατα θα είναι ήδη εγκατεστημένο στη διανομή Linux της προτίμησής σας — κι αν δεν είναι θα υπάρχει το ομώνυμο πακέτο στα αποθετήριά της. Αναλόγως του συστήματος αρχείων που θέλετε να προσαρτήσετε, ίσως χρειαστείτε και το αντίστοιχο plug-in. Αν, π.χ., επιθυμείτε να προσαρτάτε/αποπροσαρτάτε αποθηκευτικά μέσα διαμορφωμένα κατά NTFS ως απλός χρήστης, καθώς και να έχετε δικαιώματα ανάγνωσης αλλά *και* εγγραφής, το plug-in που θα χρειαστείτε είναι το NTFS-3G και θα το έχετε εγκαθιστώντας στη διανομή σας το ομώνυμο πακέτο.

Στη συνέχεια βλέπουμε πώς προσαρτούμε καταλόγους σε απομακρυσμένους servers —ασχέτως αν είναι στο τοπικό δίκτυο ή στο cloud— μέσω SSH. Η σύντομη επίδειξή μας γίνεται από σύστημα με openSUSE, παρόμοια όμως θα εργαστείτε κι από οποιαδήποτε άλλη διανομή. Χρειαζόμαστε το plug-in ονόματι SSHFS (πακέτο sshfs). Ανοίγουμε ένα παράθυρο τερματικού κι αρχικά φρεσκάρουμε την τοπική λίστα με τα περιεχόμενα των αποθετηρίων:

```
cvar@ohsuse:~> sudo zypper ref
```

Αμέσως μετά εγκαθιστούμε το SSHFS πληκτρολογώντας:

```
cvar@ohsuse:~> sudo zypper -n in sshfs
```

Αν το FUSE *δεν* είναι παρόν θα εγκατασταθεί αυτόματα, ως προαπαιτούμενο του SSHFS. Ας δούμε πώς προσαρτούμε τοπικά τον προσωπικό κατάλογο του χρήστη sub0, από τον απομακρυσμένο host με όνομα box.colder.xyz. Από τη στιγμή που η προσάρτηση γίνεται μέσω του SSH, εννοείται ότι στο απομακρυσμένο μηχάνημα υπάρχει ενεργός SSH server κι εμείς έχουμε δικαίωμα σύνδεσης στο λογαριασμό του χρήστη sub0. Φτιάχνουμε πρώτα έναν κατάλογο, ο οποίος θα αποτελέσει το σημείο προσάρτησης ή σύνδεσης για το απομακρυσμένο home directory:

```
cvar@ohsuse:~> mkdir colder
```

Είμαστε έτοιμοι για την προσάρτηση:

```
cvar@ohsuse:~> sshfs sub0@box.colder.xyz:/home/sub0 colder
```

Αυτό ήταν! Αναλόγως του αν έχουμε ρυθμίσει το public-key authentication ή όχι, δεν θα μας ζητηθεί ή θα μας ζητηθεί το password του sub0 (βλ. https://deltahacker.gr/?p=13357). Ένας τρόπος προκειμένου να βεβαιωθούμε ότι η προσάρτηση πέτυχε, είναι να γράψουμε

```
cvar@ohsuse:~> df -hT
```

Στην τελευταία γραμμή της εξόδου του df θα δούμε το απομακρυσμένο home directory, προσαρτημένο στον τοπικό κατάλογο. Ρίξτε και μια ματιά στο σχετικό screenshot.

Η αποπροσάρτηση ή αλλιώς αποσύνδεση του απομακρυσμένου καταλόγου, επιτυγχάνεται πληκτρολογώντας:

```
cvar@ohsuse:~> fusermount -u colder
```

Και μια ερώτηση για το τέλος, έτσι, για ν' ακούμε κι εμείς τις ιδέες σας: Τώρα που μάθατε για το SSHFS, πώς σκοπεύετε να το αξιοποιήσετε;

```
File  Edit  View  Search  Terminal  Help
cvar@ohsuse:~>
cvar@ohsuse:~> mkdir colder   (1)
cvar@ohsuse:~>
cvar@ohsuse:~> sshfs sub0@box.colder.xyz:/home/sub0 colder   (2)
The authenticity of host 'box.colder.xyz (46.101.173.140)' can't be established.
ECDSA key fingerprint is 61:90:0c:82:66:92:b1:c6:b9:78:19:ae:a5:5f:70:76 [MD5].
Are you sure you want to continue connecting (yes/no)? yes   (3)
sub0@box.colder.xyz's password:   (4)
cvar@ohsuse:~>
cvar@ohsuse:~> df -hT   (5)
Filesystem                     Type         Size  Used Avail Use% Mounted on
devtmpfs                       devtmpfs     982M     0  982M   0% /dev
tmpfs                          tmpfs        988M  176K  988M   1% /dev/shm
tmpfs                          tmpfs        988M  2.1M  986M   1% /run
tmpfs                          tmpfs        988M     0  988M   0% /sys/fs/cgroup
/dev/sda2                      btrfs         41G   13G   27G  33% /
/dev/sda2                      btrfs         41G   13G   27G  33% /.snapshots
/dev/sda2                      btrfs         41G   13G   27G  33% /var/lib/named
/dev/sda2                      btrfs         41G   13G   27G  33% /var/lib/mariadb
/dev/sda2                      btrfs         41G   13G   27G  33% /var/tmp
/dev/sda2                      btrfs         41G   13G   27G  33% /var/lib/pgsql
/dev/sda3                      xfs           86G  3.2G   83G   4% /home
/dev/sda2                      btrfs         41G   13G   27G  33% /var/spool
/dev/sda2                      btrfs         41G   13G   27G  33% /var/opt
/dev/sda2                      btrfs         41G   13G   27G  33% /var/lib/libvirt/images
/dev/sda2                      btrfs         41G   13G   27G  33% /var/lib/mysql
/dev/sda2                      btrfs         41G   13G   27G  33% /var/lib/mailman
/dev/sda2                      btrfs         41G   13G   27G  33% /usr/local
/dev/sda2                      btrfs         41G   13G   27G  33% /tmp
/dev/sda2                      btrfs         41G   13G   27G  33% /var/crash
/dev/sda2                      btrfs         41G   13G   27G  33% /srv
/dev/sda2                      btrfs         41G   13G   27G  33% /boot/grub2/x86_64-efi
/dev/sda2                      btrfs         41G   13G   27G  33% /opt
/dev/sda2                      btrfs         41G   13G   27G  33% /boot/grub2/i386-pc
/dev/sda2                      btrfs         41G   13G   27G  33% /var/log
sub0@box.colder.xyz:/home/sub0 fuse.sshfs    30G  7.8G   21G  28% /home/cvar/colder
cvar@ohsuse:~>
```

Για να προσαρτήσουμε με τη βοήθεια του FUSE plug-in ονόματι SSHFS τον προσωπικό κατάλογο του χρήστη sub0 από έναν απομακρυσμένο server, ξεκινάμε από τα πολύ απλά: τη δημιουργία ενός καταλόγου στο τοπικό μηχάνημα, ο οποίος θα αποτελέσει το σημείο προσάρτησης ή σύνδεσης **(1)**. Αμέσως μετά προχωράμε στην προσάρτηση, με το προγραμματάκι sshfs **(2)**. Επειδή αυτή είναι η πρώτη φορά που συνδεόμαστε στον server από τον συγκεκριμένο client, ερωτόμαστε για την αυθεντικότητα του πρώτου **(3)**. Βεβαίως, πρέπει να γνωρίζουμε και το password του χρήστη sub0 **(4)**. Η προσάρτηση επιτυγχάνει – κι ας μην πάρουμε κάποιο μήνυμα στο τερματικό. Ένας τρόπος ώστε να επαληθεύσουμε ότι πράγματι επετεύχθη, είναι παρατηρώντας την έξοδο του df **(5)**. Στην τελευταία της γραμμή φαίνεται ξεκάθαρα ότι ο απομακρυσμένος κατάλογος έχει προσαρτηθεί.

# Το Systemd κι ο έλεγχος του logging

**Για την εκκίνηση ενός συστήματος Linux υπάρχουν διάφορες μέθοδοι και μία από τις πλέον σύγχρονες, την οποία βρίσκουμε στα διαδεδομένα κι όχι μόνο distributions, υλοποιείται από το Systemd. Το Systemd κάνει πολλά περισσότερα από το να ξεκινά ένα σύστημα Linux. Κάθε διαχειριστής οφείλει να γνωρίζει τουλάχιστον τη λογική του: ακόμη κι αν δεν το χρησιμοποιεί ήδη, είναι σχεδόν βέβαιο ότι δεν θα καταφέρει να το αποφεύγει για πάντα.**

*του Χ.Β.*

Προκειμένου το Systemd να διαχειρίζεται τις υπηρεσίες του λειτουργικού συστήματος αλλά και τις καταστάσεις στις οποίες είναι δυνατόν να βρεθεί το τελευταίο, εισάγει την έννοια των στόχων (**targets**). Γενικά, με το Systemd έχουμε στόχους που πρέπει να επιτυγχάνονται, με κάθε στόχο να έχει τα προαπαιτούμενα ή αλλιώς τις εξαρτήσεις του (**dependencies**). Επιγραμματικά, κατά την εκκίνηση του λειτουργικού το Systemd θα μπορούσαμε να πούμε ότι δρα ως ακολούθως:

- Φορτώνει αρχεία ρυθμίσεων, τα οποία συνήθως βρίσκονται στον κατάλογο /etc/systemd.
- Καθορίζει ποιο είναι το λεγόμενο boot goal, μ' άλλα λόγια ποια είναι η κατάσταση στην οποία πρέπει να βρεθεί το σύστημα μετά την ολοκλήρωση της διαδικασίας εκκίνησης. Συνήθως το **boot goal** υποδεικνύεται από το αρχείο default.target (εντός του καταλόγου /etc/systemd/system), το οποίο αποτελεί symbolic link προς το αρχείο που πράγματι περιγράφει τον στόχο εκκίνησης και βρίσκεται κάτω από τον κατάλογο /usr/lib/systemd/system.
- Εντοπίζει όλες τις εξαρτήσεις που έχει το boot goal και προσπαθεί να τις ικανοποιήσει.

Όπως το παλιό System V έφερνε το λειτουργικό σύστημα σε κάποιο runlevel, ομοίως και το Systemd φέρνει το λειτουργικό σε μία κατάσταση που περιγράφεται από ένα εκ των ακόλουθων αρχείων *.target:

- **poweroff.target:** το σύστημα κλείνει (shutdown, αντίστοιχο του runlevel 0)
- **rescue.target:** το σύστημα φορτώνει σε single user mode και για την ακρίβεια μόνον ο χρήστης root μπορεί να συνδεθεί, ενώ δεν υπάρχει και δικτύωση (αντίστοιχο του runlevel 1)

- **multi-user.target:** το σύστημα φορτώνει σε multiuser mode, δηλαδή πολλοί χρήστες μπορούν να συνδέονται, ενώ υπάρχει και δικτύωση (αντίστοιχο του runlevel 3)
- **graphical.target:** το σύστημα υποστηρίζει πολλούς χρήστες, προσφέρει περιβάλλον γραφικών, υπάρχει και δικτύωση (αντίστοιχο του runlevel 5)
- **reboot.target:** το σύστημα επανεκκινεί (αντίστοιχο του runlevel 6)

Το default.target σε ένα σύστημα desktop συνήθως δείχνει στο αρχείο graphical.target (εντός του /usr/lib/systemd/system). Σε ένα σύστημα server, από την άλλη, το default.target δείχνει στο multi-user.target (ναι, επίσης εντός του /usr/lib/systemd/system). Τώρα, τα βασικά αρχεία με τα οποία το Systemd δουλεύει ονομάζονται μονάδες (**units**). Υπάρχουν units διαφόρων τύπων, κι αυτό διότι το Systemd πέρα από τη διαχείριση υπηρεσιών διεκπεραιώνει κι άλλες εργασίες όπως, π.χ., είναι η προσάρτηση συστημάτων αρχείων ή η παρακολούθηση network sockets. Επιγραμματικά, παραθέτουμε τρεις συνηθισμένους τύπους units:

- **Service units** (αρχεία με κατάληξη **.service**): Προορίζονται για την εκκίνηση υπηρεσιών (services), όπως, π.χ., είναι ο Apache ή ο OpenSSH server.
- **Mount units** (αρχεία με κατάληξη **.mount**): Χρησιμεύουν για την προσάρτηση συστημάτων αρχείων.
- **Target units** (αρχεία με κατάληξη **.target**): Πρόκειται για αρχεία που ομαδοποιούν άλλα units.

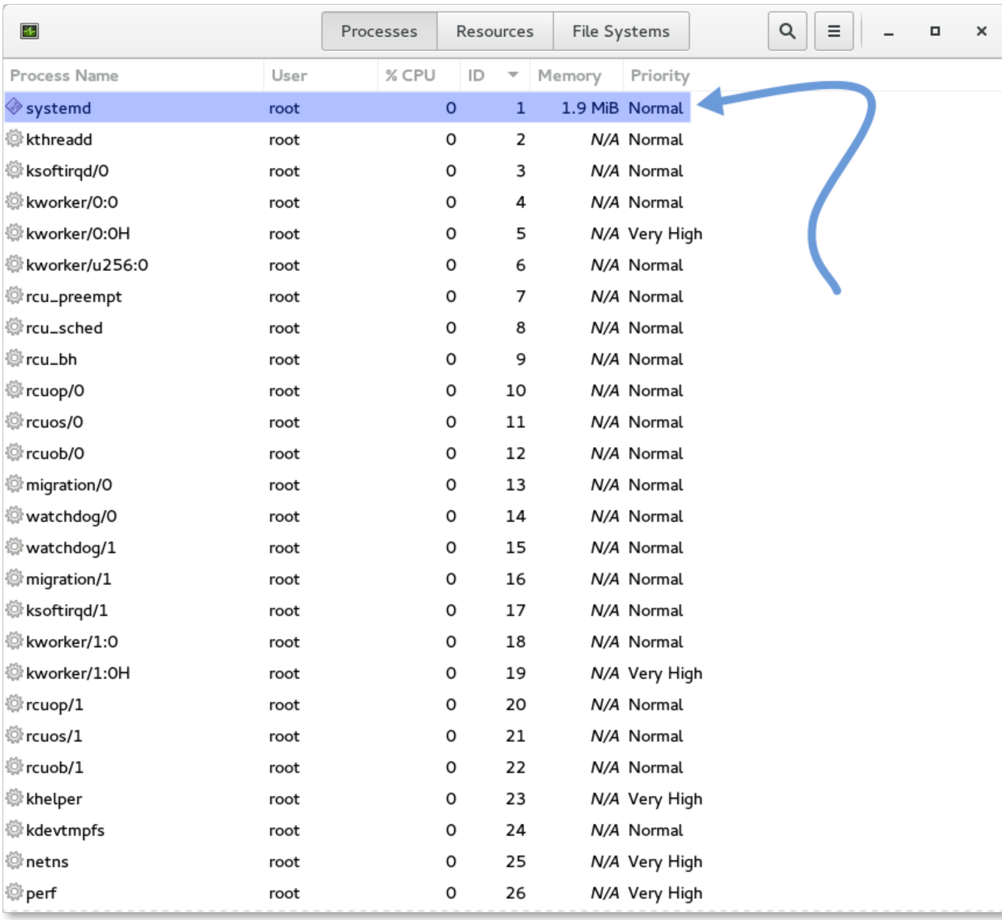
Έχουμε ήδη εισαγάγει αρκετούς νέους όρους, οπότε νομίζουμε ότι είναι ώρα για λίγη πρακτική εξάσκηση. Είναι άλλωστε ο μόνος τρόπος προκειμένου να μαθαίνουμε ουσιαστικά κι όχι απλά να στοιβάζουμε πληροφορίες.

**Σημείωση.** Για τις ανάγκες του παρόντος δουλέψαμε σε ένα σύστημα με openSUSE Leap. Εννοείται, φυσικά, ότι τους πειραματισμούς σας μπορείτε να τους κάνετε και σ' οποιαδήποτε άλλη διανομή Linux που υιοθετεί το Systemd.

## Πρώτη γνωριμία με το Systemd

Είπαμε προηγουμένως ότι το boot goal υποδεικνύεται από το αρχείο default.target, το οποίο σε ένα σύστημα desktop, όπως είναι το openSUSE Leap, εξ ορισμού δείχνει στο αρχείο graphical.target. Υποθέστε ότι θέλουμε να χρησιμοποιήσουμε το μηχάνημά μας προσωρινά ως server. Ένα τέτοιο σύστημα δεν έχει λόγο να φορτώνει το υποσύστημα γραφικών. Το ιδανικό boot goal, λοιπόν, είναι το multi-user.target. Ένας τρόπος ώστε να ξεκινήσουμε το σύστημα στο multi-user.target, είναι να περάσουμε στον πυρήνα την παράμετρο

```
systemd.unit=multi-user.target
```

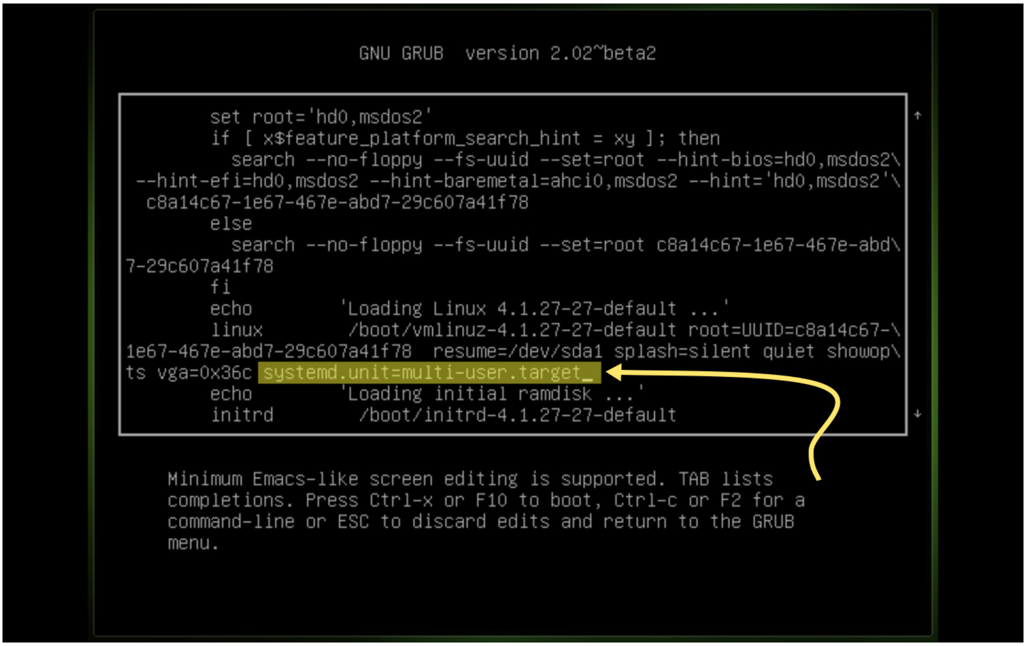


Processes | Resources | File Systems

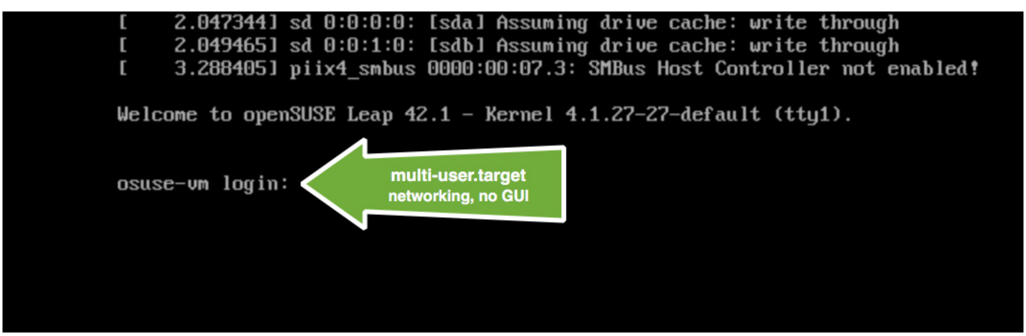
| Process Name | User | % CPU | ID | Memory | Priority |
|---|---|---|---|---|---|
| systemd | root | 0 | 1 | 1.9 MiB | Normal |
| kthreadd | root | 0 | 2 | N/A | Normal |
| ksoftirqd/0 | root | 0 | 3 | N/A | Normal |
| kworker/0:0 | root | 0 | 4 | N/A | Normal |
| kworker/0:0H | root | 0 | 5 | N/A | Very High |
| kworker/u256:0 | root | 0 | 6 | N/A | Normal |
| rcu_preempt | root | 0 | 7 | N/A | Normal |
| rcu_sched | root | 0 | 8 | N/A | Normal |
| rcu_bh | root | 0 | 9 | N/A | Normal |
| rcuop/0 | root | 0 | 10 | N/A | Normal |
| rcuos/0 | root | 0 | 11 | N/A | Normal |
| rcuob/0 | root | 0 | 12 | N/A | Normal |
| migration/0 | root | 0 | 13 | N/A | Normal |
| watchdog/0 | root | 0 | 14 | N/A | Normal |
| watchdog/1 | root | 0 | 15 | N/A | Normal |
| migration/1 | root | 0 | 16 | N/A | Normal |
| ksoftirqd/1 | root | 0 | 17 | N/A | Normal |
| kworker/1:0 | root | 0 | 18 | N/A | Normal |
| kworker/1:0H | root | 0 | 19 | N/A | Very High |
| rcuop/1 | root | 0 | 20 | N/A | Normal |
| rcuos/1 | root | 0 | 21 | N/A | Normal |
| rcuob/1 | root | 0 | 22 | N/A | Normal |
| khelper | root | 0 | 23 | N/A | Very High |
| kdevtmpfs | root | 0 | 24 | N/A | Normal |
| netns | root | 0 | 25 | N/A | Very High |
| perf | root | 0 | 26 | N/A | Very High |

*Σε κάθε διανομή Linux το systemd ξεκινάει απευθείας από τον πυρήνα, έχει process ID το 1 και αγνοεί το "θανατηφόρο" signal 9, το οποίο τερματίζει κάθε άλλη διεργασία. Όλα τα άλλα προγράμματα του συστήματος ξεκινούν είτε απευθείας από το systemd είτε από κάποια διεργασία-απόγονό του (**child process**).*

Συγκεκριμένα, στην αρχική οθόνη του GRUB επιλέγουμε τον πυρήνα προς εκκίνηση (με τα πάνω και κάτω βελάκια) και πατάμε το πλήκτρο [E] (από το Edit). Εμφανίζεται ένα παράθυρο με κείμενο, όπου βλέπουμε όλες τις οδηγίες προς τον GRUB. Με τα βελάκια του δρομέα μεταβαίνουμε στη γραμμή που αρχίζει με "linux", και καθορίζει την έκδοση του πυρήνα προς φόρτωση. Στο τέλος της γραμμής αφήνουμε ένα κενό και προσθέτουμε το "systemd.unit=multi-user.target" (χωρίς τα εισαγωγικά, βλ. και screenshot που ακολουθεί). Για την εκκίνηση του λειτουργικού πατάμε το συνδυασμό πλήκτρων [CTRL+X].




*Προσωρινή αλλαγή boot goal μέσα από τον GRUB, περνώντας στον πυρήνα την κατάλληλη παράμετρο.*




*Το σύστημα desktop των δοκιμών μας μόλις ξεκίνησε *χωρίς* περιβάλλον γραφικών. Υποστηρίζει ωστόσο δικτύωση κι επιτρέπει σύνδεση σε πολλούς χρήστες. Το boot goal του Systemd ήταν το multi-user.target, αντίστοιχο του runlevel 3 που έχει το System V.*

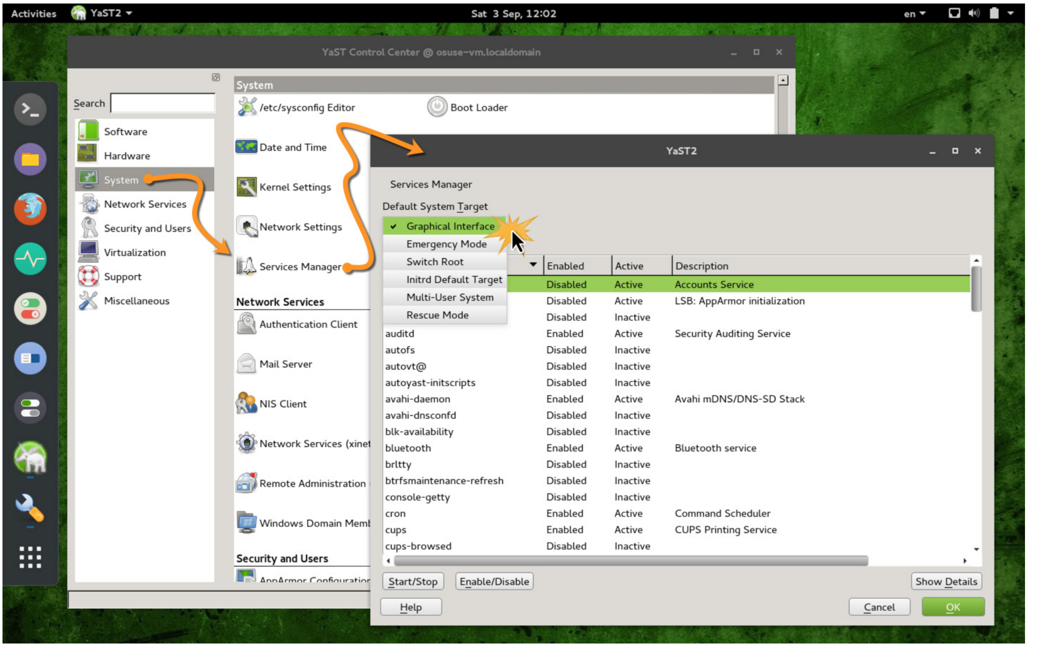
Υπενθυμίζουμε ότι η αλλαγή target με τον τρόπο που μόλις δείξαμε είναι προσωρινή, υπό την έννοια ότι στο επόμενο reboot το σύστημα θα ξεκινήσει στο προκαθορισμένο target, το οποίο για εμάς είναι το graphical.target. Εκτός από το GRUB η προσωρινή αλλαγή target γίνεται κι από τη γραμμή εντολών, χωρίς επανεκκίνηση του συστήματος. Απλά πληκτρολογούμε:

```
ajoe@osuse-vm:~> sudo systemctl isolate multi-user.target
```

Αργότερα, όταν θα θέλουμε και πάλι να επιστρέψουμε στο graphical.target, θα γράψουμε:

```
ajoe@osuse-vm:~> sudo systemctl isolate graphical.target
```

Τι κάνουμε όμως αν θέλουμε μόνιμη αλλαγή boot goal, ώστε κάθε φορά που ανοίγουμε ή επανεκκινούμε τον υπολογιστή να βρισκόμαστε στο νέο target; Κοιτάξτε, μπορούμε να πραγματοποιήσουμε την αλλαγή είτε από τη γραμμή εντολών είτε από το όποιο εργαλείο διαχείρισης για το περιβάλλον γραφικών διαθέτει η διανομή μας (στο openSUSE χρησιμοποιούμε το YaST). Συγκεκριμένα, από τη γραμμή εντολών πληκτρολογούμε:




*Αλλαγή boot goal με τη βοήθεια του YaST – και συγκεκριμένα μέσα από το module ονόματι Services Manager.*

```
ajoe@osuse-vm:~> sudo ln -sf /usr/lib/systemd/system/multi-user.target /etc/
systemd/system/default.target
```

Όπως βλέπετε, αυτό που κάναμε μόλις ήταν, απλά, να αλλάξουμε το αρχείο .target, στο οποίο δείχνει το default.target: έδειχνε στο graphical.target, τώρα δείχνει στο multi-user.target, άρα το λειτουργικό σε κάθε εκκίνηση/επανεκκίνηση θα υποστηρίζει πολλούς χρήστες και δικτύωση, χωρίς όμως να φορτώνει περιβάλλον γραφικών. Αν εξάλλου προτιμάμε τα εργαλεία για το GUI, πραγματοποιούμε αλλαγή boot goal με τη βοήθεια του YaST – και συγκεκριμένα μέσα από το module ονόματι Services Manager.

## Διαχείριση υπηρεσιών και προαπαιτούμενα στόχων

Το systemctl είναι το βασικό εργαλείο για τη διαχείριση των υπηρεσιών στον κόσμο του Systemd. Το χρησιμοποιήσαμε στην προηγούμενη ενότητα, ώστε ν' αλλάξουμε προσωρινά target. Είναι ώρα να το γνωρίσουμε καλύτερα. Το συντακτικό που καταλαβαίνει το systemctl έχει ως ακολούθως:

```
systemctl [γενικές επιλογές] υποεντολή [επιλογές υποεντολής]
```

Δίνοντας σε ένα τερματικό systemctl χωρίς τίποτε άλλο, επιστρέφονται όλα τα units που έχουν φορτωθεί. Δοκιμάστε να δώσετε την εντολή – και μην αποθαρρυνθείτε από την πληθώρα των αποτελεσμάτων. Παρατηρήστε ότι επειδή δεν χωρούν όλα σε μία οθόνη, το systemctl τα στέλνει σε έναν pager. Έτσι, με το που γεμίζει η οθόνη πατάμε [Space] για συνέχεια ή τα βελάκια του δρομέα για πάνω/κάτω ή το πλήκτρο [Q] για έξοδο. Αν πάλι δεν θέλουμε pager, δίνουμε στο systemctl τη γενική επιλογή --no-pager. Με τη δε υποεντολή list-unit-files το systemctl επιστρέφει όλα τα παρόντα στο σύστημα unit files, ασχέτως της κατάστασής τους. Και πάλι μιλάμε για πολλή πληροφορία –περισσότερη από πριν–, οπότε ίσως θέλουμε να δούμε μόνο τα unit files που αντιστοιχούν σε υπηρεσίες. Σ' αυτή την περίπτωση, στην υποεντολή list-unit-files δίνουμε την επιλογή --type=service. Συνοψίζοντας: Προκειμένου να δούμε όλες τις διαθέσιμες στο Systemd υπηρεσίες χωρίς τα αποτελέσματα να περάσουν από pager, γράφουμε:

```
ajoe@osuse-vm:~> systemctl --no-pager list-unit-files --type=service
```

Θα παρατηρήσετε ότι το παραπάνω δεν ξεχωρίζει ενεργές από μη-ενεργές υπηρεσίες και τις επιστρέφει όλες. Ένας τρόπος ώστε να δούμε μόνο τις ενεργές, είναι να δώσουμε άλλη μια επιλογή στην υποεντολή list-unit-files. Δείτε:

```
ajoe@osuse-vm:~> systemctl --no-pager list-unit-files --type=service
--state=enabled
```

Παρεμπιπτόντως, η προηγούμενη γραμμή έχει απ' όλα: εντολή (systemctl), γενική επιλογή (--no-pager), υποεντολή (list-unit-files), επιλογές υποεντολής (--type=service και --state=enabled). Μπορείτε τώρα να φανταστείτε πώς παίρνουμε μια λίστα με όλες τις υπηρεσίες που δεν είναι ενεργές; Και βέβαια μπορείτε: αντί για --state=enabled, δίνουμε --state=disabled.

```
File   Edit   View   Search   Terminal   Help
ajoe@osuse-vm:~>
ajoe@osuse-vm:~> systemctl --no-pager list-unit-files --type=service --state=enabled
UNIT FILE                                    STATE
auditd.service                               enabled
avahi-daemon.service                         enabled
bluetooth.service                            enabled
cron.service                                 enabled
cups.service                                 enabled
dbus-org.bluez.service                       enabled
dbus-org.freedesktop.Avahi.service           enabled
dbus-org.freedesktop.ModemManager1.service   enabled
dbus-org.freedesktop.nm-dispatcher.service   enabled
getty@.service                               enabled
haveged.service                              enabled
irqbalance.service                           enabled
iscsi.service                                enabled
ModemManager.service                         enabled
network.service                              enabled
NetworkManager-dispatcher.service            enabled
NetworkManager.service                       enabled
nscd.service                                 enabled
ntpd.service                                 enabled
postfix.service                              enabled
purge-kernels.service                        enabled
smartd.service                               enabled
SuSEfirewall2.service                        enabled
SuSEfirewall2_init.service                   enabled
SuSEfirewall2_setup.service                  enabled
systemd-readahead-collect.service            enabled
systemd-readahead-drop.service               enabled
systemd-readahead-replay.service             enabled
vmtoolsd.service                             enabled
wpa_supplicant.service                       enabled
YaST2-Firstboot.service                      enabled
YaST2-Second-Stage.service                   enabled

32 unit files listed.
ajoe@osuse-vm:~>
```

*Λίστα με όλες τις ενεργές υπηρεσίες του συστήματος των δοκιμών μας. Για μια λίστα με τις μη-ενεργές υπηρεσίες, στην υποεντολή list-unit-files δίνουμε την επιλογή --status=disabled.*

Αν παρατηρήσετε λίγο τη λίστα με τις ανενεργές υπηρεσίες, θα δείτε ότι μεταξύ αυτών είναι και εκείνη του OpenSSH daemon (βλ. unit file ονόματι sshd.service). Για την κατάσταση μιας υπηρεσίας της οποίας γνωρίζουμε το unit file, πληκτρολογούμε:

```
ajoe@osuse-vm:~> systemctl status sshd.service
sshd.service - OpenSSH Daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; disabled)
   Active: inactive (dead)
```

Όπως βλέπετε, το αρχείο sshd.service έχει φορτωθεί (loaded) αλλά η υπηρεσία είναι απενεργοποιημένη (disabled). Είναι επίσης και ανενεργή (inactive). Λογικά θ' αναρωτηθείτε τώρα για τη διαφορά –αν υφίσταται διαφορά– μεταξύ των όρων "απενεργοποιημένος" και "ανενεργός". Ίσως δεν φαίνεται αμέσως, αλλά διαφορά υφίσταται. Θα την εξηγήσουμε σε πολύ λίγο, αλλά ας πούμε από τώρα ότι παρόμοια διαφορά έχουμε και με τους όρους "ενεργοποιημένος" (enabled) και "ενεργός" (active). Όταν θέλουμε να ξεκινήσουμε μια υπηρεσία που είναι ανενεργή, τότε δίνουμε στο systemctl την υποεντολή start ακολουθούμενη από το όνομα του αντίστοιχου unit file. Παράδειγμα:

```
ajoe@osuse-vm:~> sudo systemctl start sshd.service
```

Δεν παίρνουμε κάποια ειδοποίηση στο τερματικό, οπότε για να βεβαιωθούμε ότι όλα πήγαν καλά γράφουμε:

```
ajoe@osuse-vm:~> systemctl status sshd.service
sshd.service - OpenSSH Daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; disabled)
   Active: active (running) since Sun 2016-09-04 08:31:32 CEST; 1min 20s ago
  Process: 2473 ExecStartPre=/usr/sbin/sshd-gen-keys-start (code=exited,
status=0/SUCCESS)
 Main PID: 2477 (sshd)
   CGroup: /system.slice/sshd.service
           └─2477 /usr/sbin/sshd -D
```

Θεσπέσια. Βλέπουμε ότι τώρα η υπηρεσία του OpenSSH είναι ενεργή (active). Παραμένει όμως απενεργοποιημένη (disabled), κι αυτή η φαινομενική ασυνέπεια φωνάζει ότι είναι ώρα να δοθούν εξηγήσεις και διευκρινίσεις.

- **Ενεργοποιημένη (enabled)** υπηρεσία: Ξεκινά κάθε φορά που ανοίγουμε ή επανεκκινούμε τον υπολογιστή, ασχέτως αν στο προηγούμενο session την είχαμε σταματήσει.

- **Απενεργοποιημένη (disabled)** υπηρεσία: Δεν ξεκινά όταν ανοίγουμε ή επανεκκινούμε τον υπολογιστή, ασχέτως αν στο προηγούμενο session την είχαμε ξεκινήσει.
- **Ενεργή (active)** υπηρεσία: Στο παρόν session τρέχει, όμως αυτό δεν σημαίνει ότι θα τρέχει και μετά από επανεκκίνηση του υπολογιστή. Για να ξεκινά αυτόματα και μετά από επανεκκίνηση, η υπηρεσία θα πρέπει να είναι *και* ενεργοποιημένη (enabled).
- **Ανενεργή (inactive)** υπηρεσία: Στο παρόν session δεν τρέχει, όμως αυτό δεν σημαίνει ότι δεν θα τρέχει και μετά από επανεκκίνηση του υπολογιστή. Για να μην τρέχει και μετά από επανεκκίνηση, η υπηρεσία θα πρέπει να είναι *και* απενεργοποιημένη (disabled).

Και μετά τα προηγούμενα, νομίζουμε ότι οι διαφορές μεταξύ των όρων enabled/active και disabled/inactive έχουν γίνει φανερές. Είδαμε εξάλλου ότι μετά την εκκίνηση του SSH daemon η υπηρεσία παραμένει disabled. Προκειμένου να είναι *και* enabled και συνεπώς να ξεκινά κάθε φορά που ανοίγουμε ή επανεκκινούμε τον υπολογιστή, η υποεντολή που δίνουμε στο systemctl είναι η enable:

```
ajoe@osuse-vm:~> sudo systemctl enable sshd.service
ln -s '/usr/lib/systemd/system/sshd.service' '/etc/systemd/system/multi-user.
target.wants/sshd.service'
```

Αυτό που μόλις έγινε, είναι ότι δημιουργήθηκε ένα symbolic link από το sshd.service του καταλόγου /etc/systemd/system/multi-user.target.wants, προς το αρχείο sshd.service του καταλόγου /usr/lib/systemd/system.

Από τις ονομασίες και μόνο κάποιων καταλόγων, αμέσως υποψιαζόμαστε τι συμβαίνει. Εν προκειμένω, το muti-user.target –όπως κι άλλα targets– έχει μια σειρά από προαπαιτούμενα. Αυτά παρατίθενται, εν είδη συμβολικών δεσμών, μέσα στον κατάλογο multi-user.target.wants. Ένας τρόπος προκειμένου να μάθουμε για όλα τα προαπαιτούμενα του multi-user.target, είναι απλά να δούμε τα περιεχόμενα του καταλόγου /etc/systemd/system/multi-user.target.wants.

Το σύστημα στο οποίο εργαζόμαστε γι' αυτή την παρουσίαση, το χρησιμοποιούμε ως desktop. Το default.target, λοιπόν, είναι το graphical.target. Μετά την ενεργοποίηση του SSH daemon, όμως, αρχίσαμε να ασχολούμαστε με το multi-user.target και τα προαπαιτούμενά του, στον κατάλογο multi-user.target.wants. Δεν θα έπρεπε να ασχοληθούμε με το graphical.target και τα περιεχόμενα του καταλόγου graphical.target.wants – ή κάτι τέτοιο; Λογική η σκέψη, όμως όπως εύκολα μπορείτε να διαπιστώσετε ο προαναφερθείς κατάλογος δεν υπάρχει εντός του /etc/systemd/system (στον οποίο κατοικεί ο multi-user.target.wants). Τι συμβαίνει εδώ; Δύο πράγματα πρέπει να λάβουμε υπόψη. Κατά πρώτον, όπως μπορούμε να διαπιστώσουμε παρατηρώντας το περιεχόμενο του αρχείου /usr/lib/systemd/system/sshd.service, η υπηρεσία του SSH "εγκαθίσταται" για το multi-user.target, αφού αυτό το target τη θέλει.

```
ajoe@osuse-vm:~>
ajoe@osuse-vm:~> sudo systemctl start sshd.service
ajoe@osuse-vm:~>
ajoe@osuse-vm:~> sudo systemctl status sshd.service
sshd.service - OpenSSH Daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; disabled)
   Active: active (running) since Sun 2016-09-04 08:39:29 CEST; 5s ago
  Process: 2553 ExecStartPre=/usr/sbin/sshd-gen-keys-start (code=exited, status=0/SUCCESS)
 Main PID: 2557 (sshd)
   CGroup: /system.slice/sshd.service
           └─2557 /usr/sbin/sshd -D

Sep 04 08:39:29 osuse-vm.localdomain systemd[1]: Starting OpenSSH Daemon...
Sep 04 08:39:29 osuse-vm.localdomain sshd-gen-keys-start[2553]: Checking for missing server keys in /etc/ssh
Sep 04 08:39:29 osuse-vm.localdomain systemd[1]: Started OpenSSH Daemon.
Sep 04 08:39:29 osuse-vm.localdomain sshd[2557]: Server listening on 0.0.0.0 port 22.
Sep 04 08:39:29 osuse-vm.localdomain sshd[2557]: Server listening on :: port 22.
ajoe@osuse-vm:~>
ajoe@osuse-vm:~> sudo systemctl enable sshd.service
ln -s '/usr/lib/systemd/system/sshd.service' '/etc/systemd/system/multi-user.target.wants/sshd.service'
ajoe@osuse-vm:~>
ajoe@osuse-vm:~> sudo systemctl status sshd.service
sshd.service - OpenSSH Daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)
   Active: active (running) since Sun 2016-09-04 08:39:29 CEST; 14s ago
 Main PID: 2557 (sshd)
   CGroup: /system.slice/sshd.service
           └─2557 /usr/sbin/sshd -D

Sep 04 08:39:29 osuse-vm.localdomain systemd[1]: Starting OpenSSH Daemon...
Sep 04 08:39:29 osuse-vm.localdomain sshd-gen-keys-start[2553]: Checking for missing server keys in /etc/ssh
Sep 04 08:39:29 osuse-vm.localdomain systemd[1]: Started OpenSSH Daemon.
Sep 04 08:39:29 osuse-vm.localdomain sshd[2557]: Server listening on 0.0.0.0 port 22.
Sep 04 08:39:29 osuse-vm.localdomain sshd[2557]: Server listening on :: port 22.
ajoe@osuse-vm:~>
```

*Ξεκινάμε την υπηρεσία του OpenSSH (1) και βεβαιωνόμαστε ότι όλα πήγαν καλά (2). Παρατηρούμε ότι η υπηρεσία, παρά το γεγονός ότι τώρα τρέχει, δεν είναι ενεργοποιημένη και συνεπώς δεν θα ξεκινήσει αυτόματα κατά την επόμενη εκκίνηση ή επανεκκίνηση του λειτουργικού. Θέλουμε η υπηρεσία να είναι διαθέσιμη ανά πάσα στιγμή, οπότε την ενεργοποιούμε (3). Μετά από έναν ακόμα έλεγχο, βλέπουμε ότι όλα είναι όπως τα θέλουμε: η υπηρεσία είναι *και* ενεργή *και* ενεργοποιημένη (4).*

```
ajoe@osuse-vm:~>
ajoe@osuse-vm:~> ls -lh /etc/systemd/system/multi-user.target.wants
total 0
lrwxrwxrwx 1 root root 38 Oct 29  2015 auditd.service -> /usr/lib/systemd/system/auditd.service
lrwxrwxrwx 1 root root 44 Oct 29  2015 avahi-daemon.service -> /usr/lib/systemd/system/avahi-daemon.service
lrwxrwxrwx 1 root root 36 Oct 29  2015 cron.service -> /usr/lib/systemd/system/cron.service
lrwxrwxrwx 1 root root 36 Jul 10 17:15 cups.service -> /usr/lib/systemd/system/cups.service
lrwxrwxrwx 1 root root 42 Jul 10 17:09 irqbalance.service -> /usr/lib/systemd/system/irqbalance.service
lrwxrwxrwx 1 root root 44 Oct 29  2015 ModemManager.service -> /usr/lib/systemd/system/ModemManager.service
lrwxrwxrwx 1 root root 46 Jul 10 17:20 NetworkManager.service -> /usr/lib/systemd/system/NetworkManager.service
lrwxrwxrwx 1 root root 36 Oct 29  2015 nscd.service -> /usr/lib/systemd/system/nscd.service
lrwxrwxrwx 1 root root 36 Jul 10 17:20 ntpd.service -> /usr/lib/systemd/system/ntpd.service
lrwxrwxrwx 1 root root 39 Oct 29  2015 postfix.service -> /usr/lib/systemd/system/postfix.service
lrwxrwxrwx 1 root root 45 Oct 29  2015 purge-kernels.service -> /usr/lib/systemd/system/purge-kernels.service
lrwxrwxrwx 1 root root 40 Oct 29  2015 remote-fs.target -> /usr/lib/systemd/system/remote-fs.target
lrwxrwxrwx 1 root root 38 Jul 10 17:15 smartd.service -> /usr/lib/systemd/system/smartd.service
lrwxrwxrwx 1 root root 36 Sep  4 08:39 sshd.service -> /usr/lib/systemd/system/sshd.service
lrwxrwxrwx 1 root root 50 Jul 10 17:20 SuSEfirewall2_init.service -> /usr/lib/systemd/system/SuSEfirewall2_init.service
lrwxrwxrwx 1 root root 45 Jul 10 17:20 SuSEfirewall2.service -> /usr/lib/systemd/system/SuSEfirewall2.service
lrwxrwxrwx 1 root root 40 Jul 10 17:09 vmtoolsd.service -> /usr/lib/systemd/system/vmtoolsd.service
lrwxrwxrwx 1 root root 46 Oct 29  2015 wpa_supplicant.service -> /usr/lib/systemd/system/wpa_supplicant.service
ajoe@osuse-vm:~>
```

*Για τα προαπαιτούμενα του multi-user.target αρκεί να δούμε τους συμβολικούς δεσμούς που περιλαμβάνονται εντός του καταλόγου /etc/systemd/system/multi-user.target.wants.*

```
File Edit View Search Terminal Help
ajoe@osuse-vm:~>
ajoe@osuse-vm:~> cat /usr/lib/systemd/system/sshd.service
[Unit]
Description=OpenSSH Daemon
After=network.target

[Service]
EnvironmentFile=-/etc/sysconfig/ssh
ExecStartPre=/usr/sbin/sshd-gen-keys-start
ExecStart=/usr/sbin/sshd -D $SSHD_OPTS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=always

[Install]
WantedBy=multi-user.target
ajoe@osuse-vm:~>
```

*Εξετάζοντας το αρχείο /usr/lib/systemd/system/sshd.service, διαπιστώσουμε ότι η υπηρεσία του SSH απαιτείται από το multi-user.target. Έτσι, μετά την ενεργοποίησή της θα είναι διαθέσιμη τόσο σ' αυτό όσο και σε κάθε άλλο target που "εμπεριέχει" το multi-user.target. Παρατηρήστε εξάλλου ότι η υπηρεσία ενεργοποιείται *μετά* το network.target. Λογικό, από τη στιγμή που έχουμε να κάνουμε με δικτυακή υπηρεσία.*

Κατά δεύτερον, το graphical.target απαιτεί την ενεργοποίηση του multi-user.target – είναι υπερσύνολό του, όπως θα έλεγε ενθουσιωδώς ένα μαθηματικός. Άρα, από τη στιγμή που η υπηρεσία του SSH ενεργοποιείται για το multi-user.target, παραμένει ενεργοποιημένη και για το graphical.target. Προκειμένου να δούμε τι απαιτεί το graphical.target, αρκεί να εξετάσουμε το περιεχόμενο του ομώνυμου αρχείου εντός του καταλόγου /usr/lib/systemd/system.

```
File Edit View Search Terminal Help
ajoe@osuse-vm:~>
ajoe@osuse-vm:~> cat /usr/lib/systemd/system/graphical.target
#  This file is part of systemd.
#
#  systemd is free software; you can redistribute it and/or modify it
#  under the terms of the GNU Lesser General Public License as published by
#  the Free Software Foundation; either version 2.1 of the License, or
#  (at your option) any later version.

[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
After=multi-user.target
Conflicts=rescue.target
Wants=display-manager.service
AllowIsolate=yes
ajoe@osuse-vm:~>
```

*Ρίχνοντας μια ματιά στο περιεχόμενο του αρχείου /usr/lib/systemd/system/graphical.target, καταλαβαίνουμε ότι το αντίστοιχο target απαιτεί το multi-user.target. Επιπρόσθετα, το graphical ενεργοποιείται *μετά* το multi-user, ενώ θέλει και την υπηρεσία του display manager. Δεν ξέρουμε πώς το βλέπετε, νομίζουμε όμως ότι το περιεχόμενο των target files συμβαδίζει με αρκετά απ' όσα γνωρίζουμε από την καθημερινή μας ενασχόληση με το Linux – τι λέτε κι εσείς;*

## Χρήσιμες υποεντολές για το systemctl

Από τη συζήτηση που ξεκινήσαμε για το systemctl και μέσα από το παράδειγμα με την υπηρεσία του SSH, έχουμε ήδη κάνει σημαντική πρόοδο στο θέμα της διαχείρισης υπηρεσιών σε ένα σύστημα Linux με Systemd. Παραθέτουμε στη συνέχεια μερικές χρήσιμες υποεντολές και τη χρησιμότητα καθεμιάς. Εκτός από μια ανάγνωση, σας προτρέπουμε να πειραματιστείτε στο δικό σας σύστημα.

**start**

Εκκίνηση υπηρεσίας για το τρέχον session. Οφείλουμε να γνωρίζουμε το όνομα του αντίστοιχου service unit. Παράδειγμα:

```
ajoe@osuse-vm:~> sudo systemctl start sshd.service
```

Αν θέλουμε η υπηρεσία να ξεκινά αυτόματα σε κάθε (επαν)εκκίνηση, τότε πρέπει να την ενεργοποιήσουμε κιόλας. Δείτε παρακάτω.

**stop**

Ομαλός τερματισμός υπηρεσίας για το τρέχον session. Παράδειγμα:

```
ajoe@osuse-vm:~> sudo systemctl stop sshd.service
```

Αν θέλουμε η υπηρεσία να μην ξεκινά αυτόματα κατά την (επαν)εκκίνηση, τότε πρέπει να την απενεργοποιήσουμε κιόλας. Δείτε παρακάτω.

**restart**

Ομαλός τερματισμός υπηρεσίας κι αμέσως μετά εκ νέου εκκίνηση. Αν η υπηρεσία δεν έτρεχε, τότε απλά ξεκινά.

```
ajoe@osuse-vm:~> sudo systemctl restart sshd.service
```

**try-restart**

Επανεκκίνηση υπηρεσίας αλλά *μόνον* αν τρέχει ήδη.

```
ajoe@osuse-vm:~> sudo systemctl try-restart sshd.service
```

**reload**

Υποχρεώνει μια υπηρεσία να διαβάσει εκ νέου τα αρχεία ρυθμίσεων, χωρίς όμως να επανεκκινήσει. Ίσως, π.χ., έχουμε τροποποιήσει το configuration file του SSH daemon, ώστε να δέχεται μόνο key-based authentication (https://deltahacker.gr/?p=13357) ή/

και για να αφουγκράζεται αιτήσεις πελατών από εναλλακτικό port (το στάνταρτ είναι το 22/TCP). Για να δουλέψει το reload πρέπει να υποστηρίζεται από την αντίστοιχη υπηρεσία.

```
ajoe@osuse-vm:~> sudo systemctl reload sshd.service
```

**reload-or-restart**

Υποχρεώνει μια υπηρεσία να διαβάσει εκ νέου τα αρχεία ρυθμίσεων. Σε περίπτωση που η υπηρεσία δεν υποστηρίζει το reload, τότε η παρούσα εντολή την επανεκκινεί. Αν η υπηρεσία *δεν* έτρεχε, τώρα ξεκινά.

```
ajoe@osuse-vm:~> sudo systemctl reload-or-restart sshd.service
```

**reload-or-try-restart**

Υποχρεώνει μια υπηρεσία να διαβάσει εκ νέου τα αρχεία ρυθμίσεων. Σε περίπτωση που η υπηρεσία δεν υποστηρίζει το reload, τότε η παρούσα εντολή την επανεκκινεί αλλά *μόνον* αν έτρεχε ήδη.

```
ajoe@osuse-vm:~> sudo systemctl reload-or-try-restart sshd.service
```

**status**

Λεπτομερείς πληροφορίες για την κατάσταση μιας υπηρεσίας. Αν η εντολή πληκτρολογηθεί με δικαιώματα root, στην έξοδό της περιλαμβάνονται και τα πλέον πρόσφατα μηνύματα που παρήγαγε η υπηρεσία.

```
ajoe@osuse-vm:~> sudo systemctl status sshd.service
sshd.service - OpenSSH Daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; disabled)
   Active: active (running) since Sun 2016-09-11 09:17:53 CEST; 12s ago
  Process: 2609 ExecStartPre=/usr/sbin/sshd-gen-keys-start (code=exited, status=0/SUCCESS)
 Main PID: 2612 (sshd)
   CGroup: /system.slice/sshd.service
           └─2612 /usr/sbin/sshd -D
Sep 11 09:17:53 osuse-vm.localdomain sshd-gen-keys-start[2609]: Checking for missing server keys in /etc/ssh
Sep 11 09:17:53 osuse-vm.localdomain systemd[1]: Started OpenSSH Daemon.
```

```
Sep 11 09:17:53 osuse-vm.localdomain sshd[2612]: Server listening on 0.0.0.0
port 22.
Sep 11 09:17:53 osuse-vm.localdomain sshd[2612]: Server listening on :: port
22.
```

**is-active**

Πληροφορία για την κατάσταση της υπηρεσίας στο τρέχον session (ενεργή ή ανενεργή). Παραδείγματα:

```
ajoe@osuse-vm:~> systemctl is-active sshd.service
active
ajoe@osuse-vm:~> systemctl is-active ModemManager.service
inactive
```

**enable**

Ενεργοποίηση υπηρεσίας, ώστε να ξεκινά αυτόματα κατά την (επαν)εκκίνηση του λειτουργικού. Αν η υπηρεσία δεν τρέχει, αυτή τη στιγμή *δεν* θα ξεκινήσει από μόνη της.

```
ajoe@osuse-vm:~> sudo systemctl enable sshd.service
ln -s '/usr/lib/systemd/system/sshd.service' '/etc/systemd/system/multi-user.
target.wants/sshd.service'
```

**disable**

Απενεργοποίηση υπηρεσίας, ώστε να μην ξεκινά αυτόματα κατά την (επαν)εκκίνηση του λειτουργικού. Αν η υπηρεσία τρέχει, αυτή τη στιγμή *δεν* θα τερματιστεί από μόνη της.

```
ajoe@osuse-vm:~> sudo systemctl disable sshd.service
rm '/etc/systemd/system/multi-user.target.wants/sshd.service'
```

**is-enabled**

Πληροφορία για το αν η υπηρεσία είναι ενεργοποιημένη ή απενεργοποιημένη.

```
ajoe@osuse-vm:~> systemctl is-enabled sshd.service
enabled
ajoe@osuse-vm:~> systemctl is-enabled ModemManager.service
disabled
```

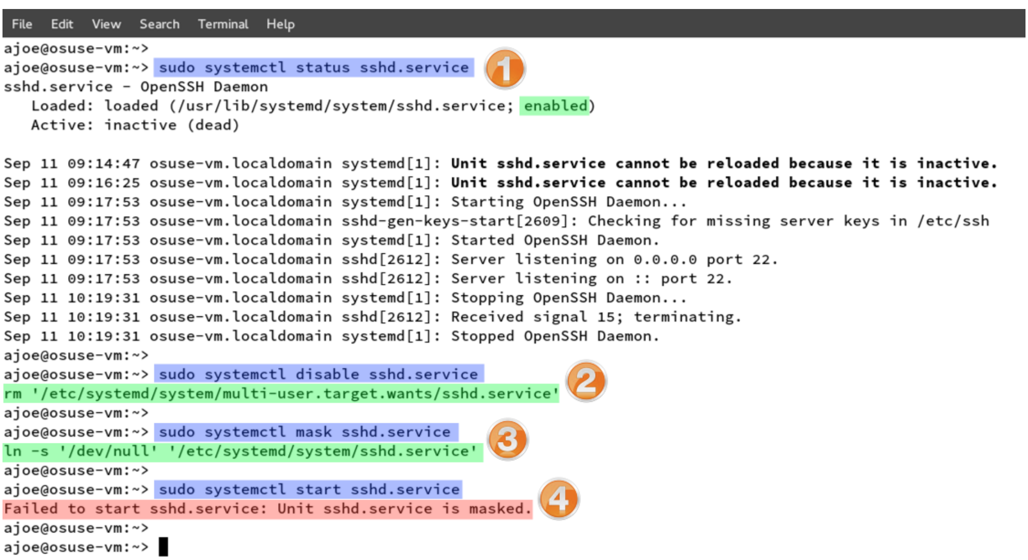
## reenable

Απενεργοποίηση υπηρεσίας κι αμέσως μετά ενεργοποίηση εκ νέου. Η εντολή είναι χρήσιμη όποτε χρειάζεται να επαναφέρουμε τα symbolic links που αφορούν στην ενεργοποίηση της υπηρεσίας, με τον τρόπο που περιγράφεται στο τμήμα "[Install]" του αντίστοιχου service unit.

```
ajoe@osuse-vm:~> sudo systemctl reenable sshd.service
rm '/etc/systemd/system/multi-user.target.wants/sshd.service'
ln -s '/usr/lib/systemd/system/sshd.service' '/etc/systemd/system/multi-user.
target.wants/sshd.service'
```

## mask

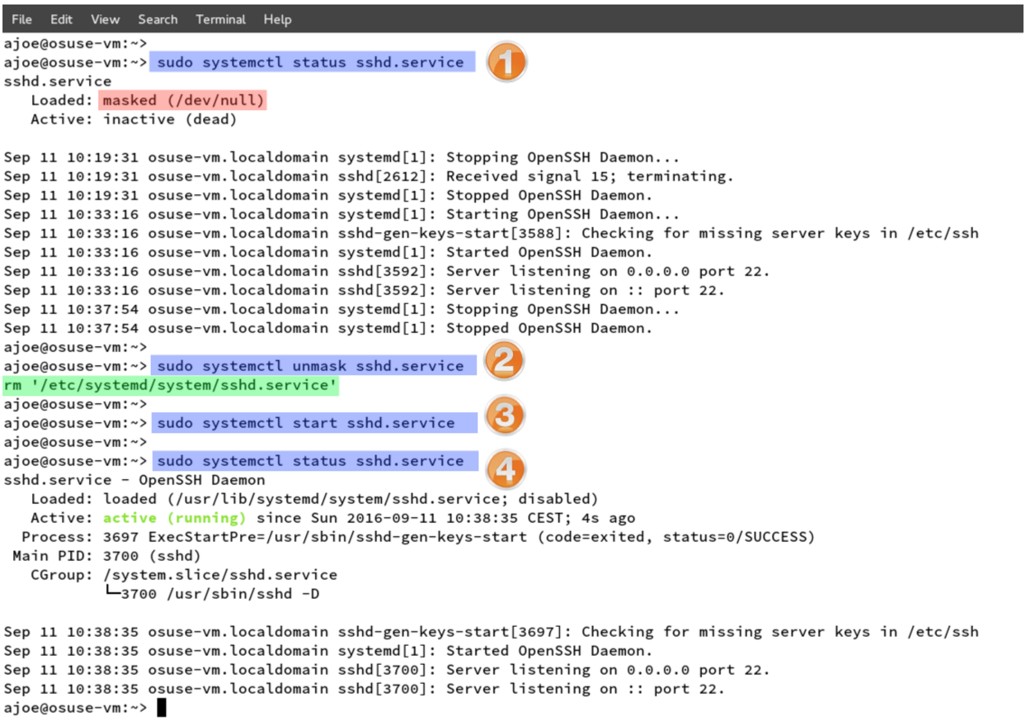
Μετά την απενεργοποίηση μιας υπηρεσίας, σε περίπτωση που δεν είναι ενεργή κάλλιστα μπορούμε να την ξεκινήσουμε χειροκίνητα. Αν όμως είναι μασκαρισμένη, τότε δεν μπορούμε.




*Έχουμε μια υπηρεσία που είναι ενεργοποιημένη και προς το παρόν ανενεργή (1). Μπορούμε φυσικά να την απενεργοποιήσουμε (2). Αν επιπρόσθετα τη μαρκάρουμε (3), τότε το χειροκίνητο ξεκίνημά της *δεν* θα είναι δυνατό (4).*

### unmask

Μία υπηρεσία που είναι μασκαρισμένη της αφαιρούμε τη μάσκα μ' αυτή την εντολή. Έτσι, μπορούμε και πάλι να την ξεκινήσουμε χειροκίνητα.



```
File  Edit  View  Search  Terminal  Help
ajoe@osuse-vm:~>
ajoe@osuse-vm:~> sudo systemctl status sshd.service
sshd.service
   Loaded: masked (/dev/null)
   Active: inactive (dead)

Sep 11 10:19:31 osuse-vm.localdomain systemd[1]: Stopping OpenSSH Daemon...
Sep 11 10:19:31 osuse-vm.localdomain sshd[2612]: Received signal 15; terminating.
Sep 11 10:19:31 osuse-vm.localdomain systemd[1]: Stopped OpenSSH Daemon.
Sep 11 10:33:16 osuse-vm.localdomain systemd[1]: Starting OpenSSH Daemon...
Sep 11 10:33:16 osuse-vm.localdomain sshd-gen-keys-start[3588]: Checking for missing server keys in /etc/ssh
Sep 11 10:33:16 osuse-vm.localdomain systemd[1]: Started OpenSSH Daemon.
Sep 11 10:33:16 osuse-vm.localdomain sshd[3592]: Server listening on 0.0.0.0 port 22.
Sep 11 10:33:16 osuse-vm.localdomain sshd[3592]: Server listening on :: port 22.
Sep 11 10:37:54 osuse-vm.localdomain systemd[1]: Stopping OpenSSH Daemon...
Sep 11 10:37:54 osuse-vm.localdomain systemd[1]: Stopped OpenSSH Daemon.
ajoe@osuse-vm:~>
ajoe@osuse-vm:~> sudo systemctl unmask sshd.service
rm '/etc/systemd/system/sshd.service'
ajoe@osuse-vm:~>
ajoe@osuse-vm:~> sudo systemctl start sshd.service
ajoe@osuse-vm:~>
ajoe@osuse-vm:~> sudo systemctl status sshd.service
sshd.service - OpenSSH Daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; disabled)
   Active: active (running) since Sun 2016-09-11 10:38:35 CEST; 4s ago
  Process: 3697 ExecStartPre=/usr/sbin/sshd-gen-keys-start (code=exited, status=0/SUCCESS)
 Main PID: 3700 (sshd)
   CGroup: /system.slice/sshd.service
           └─3700 /usr/sbin/sshd -D

Sep 11 10:38:35 osuse-vm.localdomain sshd-gen-keys-start[3697]: Checking for missing server keys in /etc/ssh
Sep 11 10:38:35 osuse-vm.localdomain systemd[1]: Started OpenSSH Daemon.
Sep 11 10:38:35 osuse-vm.localdomain sshd[3700]: Server listening on 0.0.0.0 port 22.
Sep 11 10:38:35 osuse-vm.localdomain sshd[3700]: Server listening on :: port 22.
ajoe@osuse-vm:~>
```

*Έχουμε μια υπηρεσία που είναι απενεργοποιημένη και μασκαρισμένη (1). Το χειροκίνητο ξεκίνημά της προς το παρόν δεν είναι δυνατό. Αν όμως της αφαιρέσουμε τη μάσκα (2), μπορούμε τότε να την ξεκινήσουμε (3), (4).*

## Απαιτήσεις κι επιθυμίες

Μια χρήσιμη υποεντολή του systemctl είναι η show, η οποία μεταξύ άλλων επιστρέφει τις ιδιότητες των μονάδων. Δοκιμάστε να δώσετε

```
ajoe@osuse-vm:~> systemctl --no-pager show graphical.target
```

και θα πάρετε ουκ ολίγες λεπτομέρειες για το graphical.target. Η υποεντολή show, όμως, δέχεται και κάποιες παραμέτρους που την κάνουν πολύ πιο χρήσιμη. Αν, για παράδειγμα, πληκτρολογήσουμε

```
ajoe@osuse-vm:~> systemctl --no-pager show -p "Requires" graphical.target
Requires=multi-user.target
```

τότε βλέπουμε τα προαπαιτούμενα του graphical.target (πρέπει να έχει προηγηθεί το multi-user.target). Αν πάλι πληκτρολογήσουμε

```
ajoe@osuse-vm:~> systemctl --no-pager show -p "Wants" graphical.target
Wants=display-manager.service YaST2-Firstboot.service systemd-readahead-replay.
service YaST2-Second-Stage.service systemd-readahead-collect.service systemd-
update-utmp-runlevel.service
```

τότε μαθαίνουμε για τις υπηρεσίες που είναι επιθυμητές από το graphical.target, αλλά όχι απαραίτητες.

## Διαχείριση αρχείων καταγραφής

Το Systemd είναι ικανό να διαχειρίζεται το ίδιο τα αρχεία καταγραφής (log files), συνεπώς δεν υπάρχει ανάγκη χρήσης κάποιας άλλης σχετικής υπηρεσίας (τύπου syslog). Πιο συγκεκριμένα, τα log files στον κόσμο του Systemd είναι υπό την εποπτεία και την ευθύνη του λεγόμενου journal, το οποίο δεν είναι τίποτε άλλο από μια υπηρεσία που διαχειρίζεται το ίδιο το Systemd. Το πλήρες όνομα του αντίστοιχου service unit είναι systemd-journald.service και, όπως εύκολα διαπιστώνουμε, η υπηρεσία είναι στατική (static) και ενεργοποιημένη:

```
ajoe@osuse-vm:~> systemctl status systemd-journald.service
systemd-journald.service - Journal Service
   Loaded: loaded (/usr/lib/systemd/system/systemd-journald.service; static)
   Active: active (running) since Sun 2016-09-11 14:16:19 CEST; 2h 41min ago
     Docs: man:systemd-journald.service(8)
           man:journald.conf(5)
 Main PID: 374 (systemd-journal)
   Status: "Processing requests..."
   CGroup: /system.slice/systemd-journald.service
           └374 /usr/lib/systemd/systemd-journald
```

**Σημείωση.** Το "static", στη θέση όπου θα περιμέναμε να δούμε "enabled", "disabled" ή "masked", σημαίνει ότι η αντίστοιχη υπηρεσία δεν μπορεί να ενεργοποιηθεί: Είτε κάνει μία συγκεκριμένη εργασία και μετά τερματίζεται, είτε αποτελεί προαπαιτούμενο άλλου unit και δεν πρέπει να χρησιμοποιείται από μόνη της.

Το journal συλλέγει και οργανώνει δεδομένα logging αντλώντας τα από τον πυρήνα, τις διεργασίες και τις υπηρεσίες. Αναλόγως της διανομής οι πληροφορίες καταγραφής είναι πιθανό να χάνονται κατά το reboot. Μπορούμε, βεβαίως, να επιβάλλουμε τη διατήρησή τους: με δικαιώματα root ανοίγουμε το αρχείο /etc/systemd/journald.conf και φροντίζουμε ώστε στην ενότητα [journal] να υπάρχει η γραμμή

```
Storage=persistent
```

(και φυσικά να μην είναι "σχολιασμένη"). Αποθηκεύουμε τις αλλαγές κι επανεκκινούμε την υπηρεσία του logging:

```
ajoe@osuse-vm:~> sudo systemctl restart systemd-journald.service
```

Πλέον, τα δεδομένα καταγραφής θα οργανώνονται εντός του καταλόγου /var/log/journal – και θα διατηρούνται και μετά από reboot. Αναλόγως της χρήσης που κάνουμε ή ακόμα και του ρόλου που έχει ο υπολογιστής, ο συνολικός όγκος των δεδομένων καταγραφής είναι πιθανό να ξεφύγει προς τα πάνω, δεσμεύοντας αποθηκευτικό χώρο που ίσως είναι κρίσιμος για άλλες εφαρμογές και υπηρεσίες. Πληκτρολογήστε

```
ajoe@osuse-vm:~> man 5 journald.conf
```

ώστε να δείτε πώς θα αποκλείσετε ένα τέτοιο ενδεχόμενο. Προσέξτε, π.χ., την παράμετρο SystemMaxUse, η οποία εξ ορισμού ισούται με το 10% του μεγέθους της κατάτμησης όπου αποθηκεύονται τα δεδομένα καταγραφής. Τις δυνατότητες και την ευελιξία του journal θα τις εκτιμήσουμε αφού αρχίσουμε να χρησιμοποιούμε το journalctl, το οποίο είναι το εργαλείο λήψης πληροφοριών από τα αρχεία καταγραφής. Στη συνέχεια της παρουσίασής μας δίνουμε μερικά παραδείγματα.

**Εμφάνιζε ζωντανά** (επιλογή -f) τα είκοσι πιο πρόσφατα μηνύματα του journal (παράμετρος -n 20):

```
ajoe@osuse-vm:~> sudo journalctl -n 20 -f
```

**Εμφάνισε μόνο** μηνύματα του πυρήνα (επιλογή -k):

```
ajoe@osuse-vm:~> sudo journalctl --no-pager -k
```

(αν παραλείψουμε την επιλογή --no-pager, τα αποτελέσματα εμφανίζονται με τη βοήθεια pager).

**Χωρίς τη χρήση** pager (επιλογή --no-pager), επίστρεψε τα είκοσι πιο πρόσφατα μηνύματα (παράμετρος -n 20) που αφορούν στο service unit του SSH daemon (παράμετρος -u sshd):

```
ajoe@osuse-vm:~> sudo journalctl --no-pager -n 20 -u sshd
```

**Τα μηνύματα που** αφορούν στην πιο πρόσφατη αλλά και στις προηγούμενες εκκινήσεις (boots) καταγράφονται στο journal. Προκειμένου να δούμε τα διαθέσιμα boots που έχει καταγράψει το journal, πληκτρολογούμε:

```
ajoe@osuse-vm:~> sudo journalctl --list-boots
```

**Δώστε την** παραπάνω εντολή στο σύστημά σας. Μην αποθαρρυνθείτε από το πλήθος των πληροφοριών που θα επιστραφούν. Προσέξτε μόνο τους αριθμούς στην πρώτη στήλη από αριστερά: το 0 είναι το πιο πρόσφατο boot, το -1 το προηγούμενο, το -2 εκείνο πριν το προηγούμενο κ.ο.κ. Δείτε εξάλλου ότι για κάθε boot υπάρχει και το αντίστοιχο timestamp. Αν τώρα θέλουμε τα μηνύματα του πιο πρόσφατου boot, τότε γράφουμε:

```
ajoe@osuse-vm:~> sudo journalctl --no-pager -b 0
```

**Ειδικά γι' αυτό** μπορούμε να παραλείψουμε τον αριθμό του (δηλαδή το 0). Για όλα τα προηγούμενα, όμως, τον συμπεριλαμβάνουμε υποχρεωτικά:

```
ajoe@osuse-vm:~> sudo journalctl --no-pager -b -5
```

**Εμφάνιζε ζωντανά** (επιλογή -f) όλα τα μηνύματα από τώρα και στο εξής (παράμετρος --since "now"):

```
ajoe@osuse-vm:~> sudo journalctl -f --since "now"
```

**Χωρίς pager** (επιλογή --no-pager) εμφάνισε όλα τα σημερινά μηνύματα (παράμετρος --since "today") αλλά έως τις δύο και μισή μετά το μεσημέρι (παράμετρος --until "14:30"):

```
ajoe@osuse-vm:~> sudo journalctl --no-pager --since "today" --until "13:00"
```

**Εναλλακτικά,** μια και άγρια χαράματα έχει παρατηρηθεί παράξενη συμπεριφορά στο πρόσφατο παρελθόν, με χρήση pager για δώσε λίγο όλα τα μηνύματα μεταξύ τέσσερις έως έξι, να δω κάτι:

```
ajoe@osuse-vm:~> sudo journalctl --since "04:00" --until "06:00"
```

**Δώσε όλα** τα μηνύματα που αφορούν στο χρήστη με user ID το 1000 (παράμετρος _UID=1000), ξεκινώντας από τις επτά το βράδυ (παράμετρος --since "19:00"):

```
ajoe@osuse-vm:~> sudo journalctl --since "19:00" _UID=1000
```

(Το τελευταίο ήταν παράδειγμα φιλτραρίσματος με βάση συγκεκριμένο πεδίο: το UID.)

**Κλείνουμε την** παρουσίασή μας με ένα απλό σενάριο διερεύνησης προβλημάτων περί υπηρεσιών. Ας υποθέσουμε ότι χρησιμοποιούμε τον SSH daemon και κάποια στιγμή αποφασίζουμε ν' αλλάξουμε το τυπικό port που χρησιμοποιεί η υπηρεσία: είναι το 22, αλλά για λόγους ασφαλείας θέλουμε να το αλλάξουμε στο 50022. Είχαμε διαβάσει κάτι σχετικό κάπου στο deltaHacker —δεν θυμόμαστε σε ποιο άρθρο αλλά δεν έχει σημασία— οπότε τώρα που το θυμηθήκαμε λέμε να το εφαρμόσουμε κιόλας. Με δικαιώματα διαχειριστή, λοιπόν, ανοίγουμε το αρχείο /etc/ssh/sshd_config και τροποποιούμε αναλόγως την παράμετρο Port. Βέβαιοι πως όλα πήγαν καλά, αποθηκεύουμε την αλλαγή, ξεχνάμε το όλο θέμα και συνεχίζουμε τις δουλειές μας. Αργότερα, ίσως και μια άλλη μέρα, συνειδητοποιούμε ότι είναι αδύνατον να συνδεθούμε απομακρυσμένα στο μηχάνημά μας μέσω SSH. Με το που θα βρεθούμε ξανά μπροστά του, αρχικά γράφουμε:

```
ajoe@osuse-vm:~> systemctl status sshd.service
```

Διαπιστώνουμε ότι για κάποιον λόγο η υπηρεσία δεν έχει ξεκινήσει. Με τη βοήθεια του journalctl εμφανίζουμε τις τελευταίες δεκαπέντε γραμμές του log που αφορούν ειδικά στο service unit του SSH:

```
ajoe@osuse-vm:~> sudo journalctl --no-pager -n 15 -u sshd
```

Ανάμεσα στις γραμμές βρίσκεται και η εξήγηση για την αποτυχία εκκίνησης της υπηρεσίας:

```
[...]
Sep 11 21:59:01 osuse-vm.localdomain sshd[2369]: /etc/ssh/sshd_config line 13:
Badly formatted port number.
[...]
```

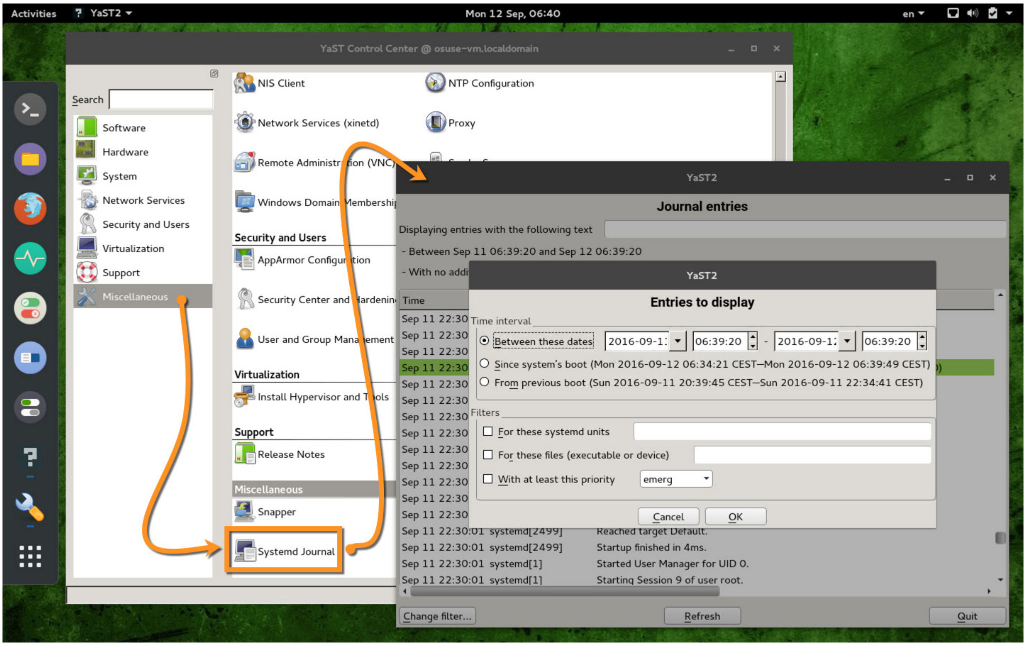
Κάτι δεν πάει καλά με το port, προφανώς: Ανοίγοντας το /etc/ssh/sshd_config βλέπουμε ότι αντί για το port 50022 είχαμε καθορίσει το (ανύπαρκτο) port 500022! Ένα μηδενικό ήταν ο υπαίτιος, αλλά χάρη στο journalctl το βρήκαμε αμέσως.

Διαβάστε το man page του journalctl για όλες τις άλλες δυνατότητές του. Επίσης: Καλωσορίσατε στον κόσμο του Systemd. Τέλος: Το Systemd έχει την ικανότητα παράλληλης εκκίνησης υπηρεσιών, παραβιάζοντας, φαινομενικά, τις μεταξύ τους εξαρτήσεις. Μπορείτε να διερευνήσετε περισσότερο, ώστε να διαπιστώσετε πώς είναι δυνατόν να καταφέρνει κάτι τέτοιο; Αν το ψάξετε, γράψτε μας τι ανακαλύψατε αφήνοντας σχόλιο κάτω από το παρόν άρθρο. Καλή σας διασκέδαση!

```
File  Edit  View  Search  Terminal  Help
ajoe@osuse-vm:~>
ajoe@osuse-vm:~> sudo systemctl status sshd.service
sshd.service - OpenSSH Daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)
   Active: failed (Result: start-limit) since Sun 2016-09-11 20:39:49 CEST; 1min 16s ago
  Process: 1281 ExecStart=/usr/sbin/sshd -D $SSHD_OPTS (code=exited, status=255)
  Process: 1277 ExecStartPre=/usr/sbin/sshd-gen-keys-start (code=exited, status=0/SUCCESS)
 Main PID: 1281 (code=exited, status=255)

Sep 11 20:39:49 osuse-vm.localdomain systemd[1]: Unit sshd.service entered failed state.
Sep 11 20:39:49 osuse-vm.localdomain systemd[1]: sshd.service holdoff time over, scheduling restart.
Sep 11 20:39:49 osuse-vm.localdomain systemd[1]: Stopping OpenSSH Daemon...
Sep 11 20:39:49 osuse-vm.localdomain systemd[1]: Starting OpenSSH Daemon...
Sep 11 20:39:49 osuse-vm.localdomain systemd[1]: sshd.service start request repeated too quickly, refusing to start.
Sep 11 20:39:49 osuse-vm.localdomain systemd[1]: Failed to start OpenSSH Daemon.
Sep 11 20:39:49 osuse-vm.localdomain systemd[1]: Unit sshd.service entered failed state.
ajoe@osuse-vm:~>
ajoe@osuse-vm:~> sudo journalctl --no-pager -n 15 -u sshd
-- Logs begin at Sun 2016-07-10 17:22:15 CEST, end at Sun 2016-09-11 20:42:00 CEST. --
Sep 11 20:39:49 osuse-vm systemd[1]: Unit sshd.service entered failed state.
Sep 11 20:39:49 osuse-vm.localdomain systemd[1]: sshd.service holdoff time over, scheduling restart.
Sep 11 20:39:49 osuse-vm.localdomain systemd[1]: Stopping OpenSSH Daemon...
Sep 11 20:39:49 osuse-vm.localdomain systemd[1]: Starting OpenSSH Daemon...
Sep 11 20:39:49 osuse-vm.localdomain sshd-gen-keys-start[1277]: Checking for missing server keys in /etc/ssh
Sep 11 20:39:49 osuse-vm.localdomain systemd[1]: Started OpenSSH Daemon.
Sep 11 20:39:49 osuse-vm.localdomain sshd[1281]: /etc/ssh/sshd_config line 13: Badly formatted port number.
Sep 11 20:39:49 osuse-vm.localdomain systemd[1]: sshd.service: main process exited, code=exited, status=255/n/a
Sep 11 20:39:49 osuse-vm.localdomain systemd[1]: Unit sshd.service entered failed state.
Sep 11 20:39:49 osuse-vm.localdomain systemd[1]: sshd.service holdoff time over, scheduling restart.
Sep 11 20:39:49 osuse-vm.localdomain systemd[1]: Stopping OpenSSH Daemon...
Sep 11 20:39:49 osuse-vm.localdomain systemd[1]: Starting OpenSSH Daemon...
Sep 11 20:39:49 osuse-vm.localdomain systemd[1]: sshd.service start request repeated too quickly, refusing to start.
Sep 11 20:39:49 osuse-vm.localdomain systemd[1]: Failed to start OpenSSH Daemon.
Sep 11 20:39:49 osuse-vm.localdomain systemd[1]: Unit sshd.service entered failed state.
ajoe@osuse-vm:~>
```

*Διερεύνηση του πραγματικού αιτίου για την αποτυχία εκκίνησης του SSH deamon. Με το systemctl διαπιστώνουμε ότι έχουμε πρόβλημα, με το journalctl βρίσκουμε σε τι οφείλεται.*




*Από το YaST, το πολυεργαλείο διαχείρισης συστήματος του openSUSE, δεν θα μπορούσε να λείπει κι ένα module για τη διαχείριση του journal. Ξέρουμε βέβαια ότι αγαπάτε την κονσόλα και τη γραμμή εντολών, κάνουμε ωστόσο και την αναφορά μας στο YaST για τους 7 αναγνώστες που προτιμούν το περιβάλλον γραφικών.*

deltahacker.gr